D 2019

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

RECONFIGURABLE FPGA-BASED BASEBAND PROCESSOR FOR MULTI-MODE SPECTRUM AGGREGATION

MÁRIO LOPES FERREIRA
PHD THESIS SUBMITTED TO
FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Reconfigurable FPGA-Based Baseband Processor for Multi-mode Spectrum Aggregation

Mário Lopes Ferreira

MAP-tele Doctoral Programme in Telecommunications

Supervisor: Prof. João Canas Ferreira

August 5, 2019

Reconfigurable FPGA-Based Baseband Processor for Multi-mode Spectrum Aggregation

Mário Lopes Ferreira

MAP-tele Doctoral Programme in Telecommunications

August 5, 2019

Abstract

The fifth generation (5G) of cellular communications will revolutionize the way humans and machines interact with each other and lead to deep technological, social and economic transformations. To fulfill to 5G potential, the communication infrastructure must support an unprecedented variety of deployment scenarios, such as enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC) or massive Machine Type Communications (mMTC). At the same time that new waveforms are studied and proposed to cope with the heterogeneity of services and requirements, the increasing ubiquity and high density of wireless communications requires a more efficient use of the electromagnetic spectrum. Consequently, the physical layer design for future wireless devices will arguably surpass those from previous generations in terms of complexity, flexibility and efficiency.

In this context, this PhD dissertation aims at studying and designing dynamically reconfigurable baseband processing architectures for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation. Due to their flexibility, parallel computation model and moderated power consumption, Field-Programmable Gate Arrays (FPGAs) are used as the implementation platform. Moreover, FPGAs allow for the specialization of computation and performance at run-time by means of Dynamic Partial Reconfiguration (DPR) and Dynamic Frequency Scaling (DFS). This work explores and evaluates these run-time reconfiguration techniques in the design of baseband processing architectures compatible with the next generation of wireless communication systems.

Considering three 5G waveform candidates - Orthogonal Frequency Division Multiplexing (OFDM), Filter-bank Multicarrier modulation (FBMC) and Universal Filtered Multicarrier modulation (UFMC) -, baseband processing datapaths for different numerologies were studied and implemented on FPGAs. In particular, the study on FBMC revealed that, despite the higher computational complexity, Frequency Spreading-FBMC modulation with Offset QAM uses less resources than an equivalent Polyphase Network-FBMC design. Analyzing the baseband processing datapaths, opportunities for run-time reconfiguration were identified and several dynamically reconfigurable baseband modulator architectures were implemented on commercial FPGAs.

A DPR-based dual-waveform modulator was implemented and compared to an equivalent static multi-mode design. Apart from reserving about half the resources used by the static multi-mode design, the DPR-based design registered an improved functional density and dynamic power savings between 13% and 52%. With a reconfiguration latency of few milliseconds and mJ-energy overhead, the application of DPR in flexible baseband processors for future wireless devices was shown to be viable, especially for commercial base stations. A parallel-pipelined OFDM modulator was also proposed. The design supports 5G New Radio numerologies and can produce 500+ MSample/s running at 160 MHz. Through DFS, the modulator clock frequency can be adapted in tens of microseconds, allowing for power savings up to 62.5%. Finally, DPR and DFS are combined into a baseband processing architecture featuring three independent modulators. Each modulator can be dynamically reconfigured to process non-contiguous component carriers using different waveforms or numerologies. Using DPR, the overall circuitry implemented on the architecture exceeds the slices available on a xc7z020 FPGA device by 17%. The proposed architectures are shown to be highly flexible, scalable, cost-effective and forward-compatible.

This work contributes to the study and design of hardware infrastructures for future communications and finds a particular application in centralized baseband processing units for 5G Cloud-RAN systems.

Resumo

A quinta geração (5G) de telecomunicações móveis irá revolucionar a forma como humanos e máquinas interagem entre si e conduzir a profundas transformações tecnológicas, sociais e económicas. Para materializar o potencial do 5G, a infraestrutura de comunicações deverá suportar uma variedade de cenários de aplicação sem precedentes, tais como banda larga móvel aumentada, comunicações ultra fiáveis de baixa latência ou comunicações tipo-máquina massivas. Ao mesmo tempo que novas formas de onda são estudadas e propostas para lidar com a heterogeneidade de serviços e requisitos, a ubiquidade e elevada densidade das comunicações móveis requer uma utilização mais eficiente do espectro eletromagnético. Deste modo, o desenvolvimento da camada física para os dispositivos sem fios do futuro irá indiscutivelmente ultrapassar os dispositivos de gerações anteriores em termos de complexidade, flexibilidade e eficiência.

Neste contexto, esta dissertação de doutoramento procura estudar e projetar arquiteturas dinamicamente reconfiguráveis para processamento em banda base adequadas a comunicações multi-modo, à coexistência de múltiplas formas de onda e à agregação de espectro dinâmica. Devido ao seu modelo de computação paralela, flexibilidade e consumo energético moderado, *matrizes de elementos lógicos programáveis* (FPGAs) são utilizadas como plataforma de implementação. Além disso, as FPGAs possibilitam a especialização da computação e do desempenho em tempo de execução, através da *reconfiguração parcial dinâmica* (DPR) e *ajustamento de frequência dinâmico* (DFS). Este trabalho explora e avalia estas técnicas de reconfiguração em tempo de execução no desenvolvimento de arquiteturas para processamento em banda base compatíveis com a próxima geração de sistemas para comunicações sem fios.

Considerando três potenciais formas de onda para 5G - *Orthogonal Frequency Division Multiplexing* (OFDM), *Filter-bank Multicarrier modulation* (FBMC) e *Universal Filtered Multicarrier modulation* (UFMC) -, processadores banda base foram estudados e implementados em FPGA, utilizando diferentes numerologias. Em particular, o estudo efetuado para FBMC revelou que, apesar da maior complexidade computacional, a modulação *Frequency Spreading*-FBMC com *Offset* QAM usa menos recursos do que um modulador equivalente do tipo *Polyphase Network*-FBMC. Através da análise dos processadores banda base, foram identificadas oportunidades para reconfiguração em tempo de execução e diversas arquiteturas dinamicamente reconfiguráveis para moduladores banda base foram implementadas em FPGAs comerciais.

Um modulador *dual-waveform* baseado em DPR foi implementado e comparado com um modulador multi-modo estático equivalente. Além de reservar cerca de metade dos recursos utilizados pelo modulador multi-modo estático, o modulador baseado em DPR registou uma melhor densidade funcional, bem como reduções entre 13% a 52% no consumo dinâmico. Uma latência de reconfiguração de poucos milissegundos e um acrescento de consumo na ordem dos milijoule, demonstra a viabilidade da aplicação de DPR em processadores banda base flexíveis para sistemas sem fios do futuro, especialmente em estações base. Um modulador OFDM *parallel-pipelined* foi também proposto. O modulador suporta numerologias propostas para 5G *New Radio* e é capaz de produzir mais de 500 MAmostra/s, operando a 160 MHz. Através de DFS, a frequência de operação do modulador pode ser adaptada em dezenas de microssegundos, possibilitando reduções de consumo até 62.5%. Por fim, DPR e DFS foram combinadas numa arquitetura de processamento banda base que contém três moduladores independentes. Cada modulador pode ser dinamicamente reconfigurado para processar bandas de espectro não contíguas utilizando diferentes formas de onda e numerologias. Aplicando DPR, o conjunto de circuitos implementados na arquitectura excede em 17% as recursos lógicos disponíveis no dispositivo xc7z020. As arquiteturas propostas demonstram ser altamente flexíveis, escaláveis, rentáveis e compatíveis com atualizações futuras.

Este trabalho contribui para o estudo e desenvolvimento de infraestruturas de hardware para as comunicações futuras e encontra uma aplicação prática em unidades centralizadas de processamento banda base para sistemas 5G *Cloud-RAN*.

Acknowledgements

I would like to acknowledge and thank the support provided by *FCT - Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology), through the Ph.D. Grant *PD/BD/105860/2014* and the Supplementary Training Grant *CRM:0067654*.

I would also like to show my gratitude to the *Instituto de Engenharia de Sistemas e Computadores* (INESC TEC) and the *Faculdade de Engenharia da Universidade do Porto* (FEUP) for providing me all the necessary material and facilities to develop and publish my research work.

On a personal level, my special thanks to Prof. João Canas Ferreira for the support, confidence, patience and knowledge. His insightful guidance and encouragement were crucial throughout the course of the last four years and indelibly marked my Ph.D dissertation and related research.

Besides my supervisor, I would like to express my gratitude to Prof. Michael Hübner for the opportunity to collaborate with his team during 5 months at the *Chair for Embedded Systems for Information Technology* (ESIT) at the *Ruhr-Universität Bochum* (RUB). I felt a very warm welcome by everyone and the easy integration within the team resulted in a pleasant and fruitful stay.

My sincere gratitude to all my colleagues and peers at FEUP, INESC TEC and ESIT for the support and stimulating discussions. In particular, I am grateful to Amin Barahimi, Benedikt Janßen, Miguel Carvalho and Nuno Paulino.

Last but certainly not the least, I would like to express my profound gratitude to my family and friends. To my parents, Clara and Domingos, for their unconditional, constant love and encouragement. They are the cornerstones of everything I am and have achieved so far. To my sister, Joana, for her affection and tenderness. To my grandparents, Maria Joaquina, Luís, Maria Celeste and José Zacarias, who I truly admire and whose values and intangible wisdom have shaped my personality. To Ana, who joined me during this journey and was always there to support me and to alleviate the burden of finishing this Ph.D dissertation.

Mário Lopes Ferreira

"O que faz andar a estrada? É o sonho.
Enquanto a gente sonhar a estrada permanecerá viva.
É para isso que servem os caminhos,
para nos fazerem parentes do futuro."

"What is it that makes the road move along? It's our dreams.
As long as people dream, the road will stay alive.
That's what the roads are for,
to make the future our kin."

Mia Couto

Contents

List of Figures

List of Tables

List of Abbreviations

2G	2^{nd} generation (5G) of cellular communications
3G	3^{rd} generation (5G) of cellular communications
3GPP	3^{rd} Generation Partnership Project
4G	4^{th} generation (5G) of cellular communications
5G	5^{th} generation (5G) of cellular communications
ADT	All-Digital Transceiver
ASIC	Application Specific Integrated Circuit
CA	Carrier Aggregation
CLB	Configurable Logic Block
CP	Cyclic Prefix
CR	Cognitive Radio
DMA	Direct Memory Access
DPR	Dynamic Partial Reconfiguration
DSP	Digital Signal Processor
eMBB	Enhanced Mobile Broadband
f-OFDM	Filtered Orthogonal Frequency-Division Multiplexing
FBMC	Filter-bank Multicarrier
FCC	Federal Communications Commission
FF	Flip-flop
FFT	Fast Fourier Transform
FIFO	First-In, First-Out
FIR	Finite Impulse Filter
FPGA	Field-Programmable Gate Array
GFDM	Generalized Frequency Division Multiplexing
GPP	General Purpose Processor
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IMT-2020	International Mobile Communications-2020 standard
IOB	Input/Output Block
ITU	International Telecommunication Union
ITU-R	ITU Radiocommunication Sector
LAN	Local Area Network
LDPC	Low-Density Parity-Check
LUT	Lookup Table

MAC	Media Access Control
MIMO	Multiple-input and multiple-output
mMTC	Massive machine type communications
NBI	Narrow Band Interference
NC-OFDM	Non-Contiguous Orthogonal Frequency Division Multiplexing
NoC	Network on Chip
NR	New Radio
OFDM	Orthogonal Frequency Division Multiplexing
OOB	Out-of-Band Interference
PAPR	Peak-to-Average Power Ratio
PHY	Physical Layer
PoC	Proof of Concept
P&R	Place and Route
QoS	Quality of Service
RAM	Random Access Memory
RC	Reconfigurable Computing
RM	Reconfigurable Module
RP	Reconfigurable Partition
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SRAM	Static Random-Access Memory
UFMC	Universal Filtered Multicarrier
UWB	Ultra-Wideband
URLLC	Ultra-Reliable and Low Latency Communications
V2X	Vehicle-to-Everything
WLAN	Wireless Local Area Network

Chapter 1

Introduction

In this introductory chapter, the research activities developed within this dissertation are contextualized and motivated. Next, the main goals sustaining this dissertation are presented. In the end of this chapter, the structure of the present document is described.

1.1 Context

The evolution of cellular communications has been marked by higher demands for reliable, uninterrupted global coverage as well as a wide range of resource consuming applications and services. As a consequence, the number of communication standards and protocols has increased, following the rapid and continuous developments in the telecommunications field. A flagrant example of this phenomenon is the variety of protocols - for cellular network (2G, 3G and 4G), WiFi, GPS reception and Bluetooth - currently incorporated into consumer products, such as smart-phones, tablets or smart-watches. It is unlikely that this trend will abate, as we approach the realization and full deployment of the 5^{th} generation (5G) of cellular communications. Although previous generations were important drivers of social and technical evolution, 5G promises a bigger impact in the way humans and machines interact by optimizing existing applications and enabling numerous innovative solutions in new application fields, such as transportation, education or medical science. The economic impact of 5G is likely to be massive. A report by IHS Markit, a global information provider, estimates that, in 2035, the 5G value chain will reach $3.5 trillion in revenue and support about 22 million jobs, while the global revenue of a wide range of 5G-enabled industry sectors may be $12.3 trillion (Campbell et al., 2017).

This revolution is well demonstrated by the three main types of 5G use cases and services defined by the International Telecommunication Union (ITU): enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC) and massive Machine Type Communications (mMTC) (ITU-R, 2015). Viewed as an evolution of 4G mobile broadband services, eMBB targets human-centric use cases characterized by very high peak and user experienced data rates (10^9 bit/s and 50+ Mbit/s, respectively), large bandwidth and increased system capacity. Pervasive video, broadcast access everywhere and every time or seamless 3D video and UHD screen sharing in

crowded events are some examples of eMBB scenarios. URLLC use cases require low latency (below 1 ms), high reliability (over 99.999%) and availability. For instance, mission critical device communications in Industry 4.0 (Wollschlaeger et al., 2017) or vehicle-to-everything (V2X) networks (Boban et al., 2016), as well as real-time human-machine interaction through Tactile Internet (Simsek et al., 2016) in health-care services (Soldani et al., 2017) can require this type of communications. mMTC use cases are characterized by low bandwidth, infrequent and low volume data transmissions between densely connected, low-power devices (10^6 devices per km^2). Massive sensor networks in Internet-of-Things (IoT) for Smart Cities are a key example for this type of use cases (Cia et al., 2018).

As a consequence, the physical layer (PHY) for the next generation of wireless systems will be far more complex than that of its predecessors. From a high-level perspective, the PHY wireless communication system can be decomposed into six sections: antenna system, radio front-end, analog/digital conversion, digital up/down-conversion, baseband processing and data processing. This dissertation addresses the baseband processing domain, where signal synthesis/analysis is performed according to the required waveform design. More emphasis was put on waveform specific operations. The baseband modulator receives groups of bits from a Forward Error Correction (FEC) coder, converts them into complex I/Q samples and processes them to produce a time-domain symbol that is forwarded to the digital up-converter. On the other side, the baseband demodulator receives a time-domain symbol from the digital down-converter and processes it into sets of bits that are fed to the FEC decoder. Forward Error Correction operations are dealt in the data processing section and decimation/interpolation operations are done in the digital up/down-conversion section.

The waveform is the cornerstone of baseband processing in wireless devices. *Orthogonal Frequency-Division Multiplexing* (OFDM) was the preferred waveform in 4G and the 3GPP Release 15 (TS, 2018) recently defined it as the multiple access scheme for the 5G New Radio (NR) PHY. This is specially due to its high frequency selectivity, flexibility, efficient hardware implementation by FFT/IFFT modules and good Multiple-Input Multiple-Output (MIMO) compatibility (Andrews et al., 2014). Despite these advantages, the spectrum of OFDM symbols presents large sidelobes that cause high out-of-band (OOB) emissions. Moreover, the interference between adjacent time-domain symbols is mitigated by adding redundancy to each symbol, which reduces the spectral efficiency. These drawbacks can hinder the achievement of 5G requirements in certain communication scenarios. For these cases, several other waveforms have been proposed (Luo and Zhang, 2016). Among these waveforms, the most popular are *Filter-bank Multicarrier* modulation (FBMC), *Universal Filtered Multicarrier* modulation (UFMC), Filtered OFDM (f-OFDM) or *Generalized Frequency Division Multiplexing* (GFDM). Obviously, changing the waveform will also change the operations involved in baseband processing. Specially for sub-6 GHz spectrum bands, the *coexistence* of multiple numerologies and waveforms, as well as the tight interworking between 5G and previous generations will be mandatory and likely to happen in the near-future (Jue, 2017).

The massification and omnipresence of wireless communication raises concerns about the overcrowding and inefficient use of the electromagnetic spectrum. Apart from expanding the

spectrum utilization to currently unused frequency bands (above 6 GHz), a more efficient spectral utilization of heavily used bands is a critical objective. To tackle this issue, future baseband processor designs should support *dynamic spectrum access* (DSA) (Zhao and Sadler, 2007) and *carrier aggregation* (CA) schemes. DSA advocates the use of vacant portions of the spectrum - spectrum holes - by unlicensed users without causing interference with licensed users. Through carrier aggregation, these spectrum holes are combined to improve spectrum efficiency and system capacity. Introduced in 4G systems, carrier aggregation will also be crucial to achieve better robustness and peak performance in 5G systems (Bhushan et al., 2017). There are two types of CA techniques: contiguous CA and non-contiguous CA. In the former case, aggregated component carriers are adjacent to each other, whereas in the later case, aggregated component carriers are spread across different spectrum bands. Although contiguous CA does not require deep changes in the transceiver baseband architecture, multidimensional and flexible baseband processing blocks are required for non-contiguous CA, in order to adaptively tune communication parameters for each component carrier to aggregate (Yuan et al., 2010).

From the scenarios just presented, baseband processing infrastructures for future wireless devices must be:

- *1) flexible* to adapt their operation for different communication setups (i.e. waveforms and their variable parametrization);
- *2) scalable* to tune performance and capacity according to communication demands;
- *3)* resource and power *efficient* for cost-effectiveness and *green communication* (Akyildiz et al., 2016) purposes;
- *4) forward-compatible* to easily integrate the support for new services and requirements, extending system lifetime.

Proposed by Joseph Mitola, the Software Defined Radio (SDR) (Mitola, 1995) concept is well-suited to these requirements. An SDR consists of a flexible radio device whose PHY supports multiple waveforms and modes of operation that are dynamically controlled by software. In an ideal SDR, the baseband processing would be implemented in software (SW), using a Digital Signal Processor (DSP) or a General-Purpose Processor (GPP). However, baseband processing operations often involve high computational requirements that are beyond the software capacity. Thus, in practical SDR systems, these operations are defined by software but actually implemented in hardware (HW). In this regard, the design of hardware infrastructures for baseband processing that provide high computational performance without compromising flexibility and power efficiency is a very important challenge (Zhang et al., 2016).

Flexibility itself can be achieved by integrating on a single chip a wide variety of task-specific hardware blocks for handling every supported mode of operation. Then, multiplex-based circuitry would be used to choose the necessary system configuration according to the current communication demands. This approach is possible due to the high levels of circuit integration enabled by the Moore's Law and current manufacturing technology. Alternatively, if the operations to perform

present some computational and/or structural regularity, a hardware block for the most demanding mode of operation could be designed and dynamically adjusted to different modes of operation by setting the value of certain operational parameters. An advantage of these approaches would be the high reconfiguration speed. However, such extensive systems would not be permanently needed, as during most periods some hardware blocks would not be used. These approaches would then exhibit an inefficient use of the available resources. Moreover, the unused hardware blocks would still consume power, since they would continue to belong to the active circuit. Also, if upgrades are needed in order to extend the functionality of a certain module, a complete design has to be produced. These problems can be overcome with *Reconfigurable Computing* (RC): using spatially programmable architectures to perform computations (Tessier et al., 2015). The concept of programmable hardware is inherent to RC and attempts to blend the high-performance and low-power from Application-specific integrated circuits (ASICs) with the flexibility from GPPs (Hsiung et al., 2009).

Since the mid-1980s, RC has gained a significant popularity, specially due to the introduction of Field-Programmable Gate Arrays (FPGAs) - semiconductor devices composed of logic cells and interconnects that can be reprogrammed for a given application or functional requirements after manufacturing. Steve Trimberger defines three ages for the FPGA technology (Trimberger, 2015) so far: the *age of invention* (1984-1991), the *age of expansion* (1992-1999) and the *age of accumulation* (2000-2007). The age of invention started with the launching of the first commercial FPGA: the Xilinx xc2064. During this age, FPGAs could not compete with ASICs in terms of high performance and capacity. Still, FPGAs were mainly used for glue-logic implementation and proved to be a good solution for less demanding applications, due to their lower up front non-recurring engineering cost and reduced production risk. The progress of Moore's Law in the 1990's was crucial for FPGA technology improvement and affirmation – the *age of expansion*. As the amount of transistors doubled every two years, foundries looked at FPGAs as a good target to launch and polish their new fabrication processes. This considerably improved the capacity and performance of FPGAs, that could now present themselves as a competitive solution in many of ASIC application domains.

The continuous pace of the Moore's Law led to FPGAs with more capacity than required by most applications. Thus, FPGA vendors focused on the accumulation and integration of new dedicated blocks with the pre-existing LUTs, FFs, IOBs and interconnects - *age of accumulation*. These new blocks included memories, digital signal processing slices, microprocessors, high-speed transceivers, among others. Currently, rather than programmable logic, state-of-the-art FPGAs have evolved towards a fully programmable System-on-Chip (SoC) that enables advanced HW/SW co-design and comprises a programmable logic block tightly interconnected with memory, mixed-signal interfaces for temperature and voltage monitoring and an on-chip network. In the communications domain, the flexibility and parallel computation of FPGAs make them specially suitable for base stations (Moy and Palicot, 2015). Nonetheless, FPGAs are currently used in a broad class of other domains like financial, data-center architectures, machine learning, scientific computing or even molecular dynamics (Tessier et al., 2015).

1.2 Motivation and Problem Statement

FPGAs are intrinsically reconfigurable: the device will perform a new functionality, once a new bitstream is loaded into the FPGA configuration memory. In other words, FPGA-based reconfigurability can be viewed as a "*software-defined functionality, where flexibility is controlled predominately through the specification of bit patterns*" (Lyke et al., 2015). The traditional way to alter the functionality of an FPGA consists of downloading a new bitstream onto the configuration memory and rebooting the system. The reconfiguration process is independent of the application (Hsiung et al., 2009) and it is designated by *static reconfiguration*. Alternatively, SRAM-based FPGAs allow the reconfiguration process to be part of the application, such that it occurs on-the-fly during execution time in response to a certain system functional requirement, without turning off the FPGA. This process is known as *dynamic reconfiguration*. FPGA reconfiguration can also be classified into complete and partial. If the bitstream loaded in the FPGA includes information about the whole device, the reconfiguration is complete. But it is also possible to change only some portions of the design without reconfiguring the whole chip area - *partial reconfiguration*. In certain applications, it would be desirable to reconfigure portions of the device without completely turning it off, that is, while it is still operating - *dynamic partial reconfiguration* (DPR) (Vipin and Fahmy, 2018).

In DPR, parts of the programmable logic are designated to be reconfigured during run-time - reconfigurable partitions (RPs). The functionality of the RPs can be changed while the static part of the system continue to operate. For a given RP functionality, a partial bitstream is generated. As opposed to complete bitstreams, the size of partial bitstreams depends on the circuit area to be reconfigured. For a given RP, there are several reconfigurable modules (RMs) associated with it. At a certain instant, only one RM occupies a certain RP. For each RM, a partial bitstream has to be produced. In terms of design flow, an RP has to be correctly sized and placed, in order to be assigned the resources required by the most demanding RM that can operate in that RP. The enhanced DPR flexibility allows for the "*specialization of the computation to the near-instantaneous needs of the application*" (Tessier et al., 2015).

Furthermore, other interesting DPR features are pointed out in (Crockett et al., 2014). As the amount of RMs is only limited by the available storage capacity for partial bitstreams, DPR enables a big variety of functionalities - *feature wealth*. Moreover, the addition of new functionalities only requires the creation of new partial bitstreams, instead of the complete system redesign. This contributes to an easy *system upgradeability*. The time-multiplexing of mutually-exclusive RP functionalities reduces circuit area and increases functional density. Ultimately, this *hardware virtualization* can allow the implementation of a collection of circuits whose aggregated amount of resources exceeds the FPGA capacity. Consequently, a device with smaller hardware capacity can be used for the target system implementation, with immediate benefits in terms of *cost-effectiveness* and resource and power *efficiency*. Regarding power efficiency, Liu et al. (Liu et al., 2009) state that DPR can reduce both dynamic and static power, as opposed to clock gating, which reduces only dynamic power. These benefits are aligned with the features desired for baseband

processors mentioned earlier in Section 1.1. In fact, FPGAs and DPR have been pointed out as suitable platforms and design technique to implement baseband processing architectures for flexible wireless systems (Lyke et al., 2015; Vipin and Fahmy, 2018; Crockett et al., 2014; Moy and Palicot, 2015). Moy and Palicot (Moy and Palicot, 2015) even suggest that the advent of SDR was a catalyst for the development of the DPR technology by Xilinx. Despite the aforementioned benefits, DPR also entails overheads related with reconfigurable area reservation, additional memory requirements to store partial bitstreams, reconfiguration time and reconfiguration energy.

The impact of the reconfiguration times is of particular importance in real-time communication systems, as it can affect the system responsiveness and the overall Quality-of-Service (QoS). The baseband processing reconfiguration can occur during the setup of a new communication (i.e. before any data transfer) or during the communication process, due to changes in the communication environment (e.g. channel quality, coverage, Signal-to-Noise Ratio (SNR), available bandwidth). In the former case, the reactivity requirements are less strict than in the later case. ITU-R has recently released a report on the minimum performance requirements for IMT-2020 communications (ITU-R, 2017). Two of the requirements are related with the control plane latency and mobility interruption time. The control plane latency is the time associated with the transition from an inactive, energy-efficient state (idle state) to the start of continuous data transfer (active state). According to the ITU-R report, this latency should be less than 20 ms. System designers are even encouraged to further reduce it to 10 ms. The reconfiguration time for baseband processors should be shorter than these values because there are other tasks to be performed during the control plane latency period. The mobility interruption time is the longest allowed time interval during which a device cannot exchange data with any base station, due to handover procedures. In eMBB and URLLC scenarios, the minimum requirement for mobility interruption time is 0 ms. This means that *make-before-break* approaches and multi-connectivity solutions must be adopted in order to setup a new connection before dropping the old one (Marsch et al., 2018).

Apart from system responsiveness, the QoS will depend strongly on the processing throughput offered by the radio system. The OFDM numerologies from (TS, 2018) require sampling rates up to 491.52 MHz, which is 16 times more than the most demanding 4G LTE sampling rate for downlink transmission. Although possible in state-of-the-art FPGAs (Ahmadi, 2016), it becomes very challenging to push the clock frequency to 500 MHz. In several use cases, such high clock frequencies are not permanently required, leading to system over-optimization and inefficient power consumption. So, apart from flexibility and system responsiveness concerns, it is also important to study and evaluate baseband processor architectures with a balanced performance-power consumption relation.

The application of DPR in baseband processing systems is not new. The first attempts applied DPR to small-scale, relatively simple and isolated modules, rather than providing an integrated solution for a complete baseband processing datapath. More recently, several works adopted DPR techniques to implement flexible baseband modulators. Nevertheless, previously published work neglects a proper evaluation of DPR overheads in this application domain. There is also a lack of extensive and fair head-to-head *DPR vs. Static* approaches comparison, considering functionally

equivalent architectures. To date, there is no architectural proposal for a baseband processor flexible and scalable enough to be deployed in dynamic spectrum aggregation scenarios. Starting from the hypothesis that DPR provides a convenient architectural concept to design baseband processing engines, this dissertation addresses the following problem:

How to design FPGA-oriented baseband processing architectures compatible with the next generation of wireless communication systems?

1.3 Aim and Scope

In order to answer to the formulated problem, the aim of the present dissertation is *the investigation and design of dynamically reconfigurable baseband processing architectures for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation*. It is worth mentioning that the focus of this research will be on the reconfigurable architecture concept rather than on the implementation details of each individual baseband processing operation. The idea is to design baseband processing engines able to dynamically reconfigure their operation, performance and capacity in response to communication demands. Obviously, this will require interventions on the baseband processing modules. In this regard, a special attention will be given to the FFT/IFFT operations. They are fundamental blocks in the transceiver design for the most relevant multicarrier waveforms and their considerable computational complexity requires efficient hardware implementations.

Apart from OFDM, two other 5G waveform candidates are considered: FBMC and UFMC. Still, additional emphasis is put on OFDM with the design of a datapath for baseband modulation (transmitter side) and demodulation (receiver side). For FBMC and UFMC, only baseband modulation (transmitter side) is studied. In general, the proposed reconfigurable architectures are mainly focused on baseband modulation (transmitter side) and were implemented on commercial Xilinx FPGAs.

1.4 Contributions

This dissertation extends the state of the art with several contributions. The main state of the art advancement directly targets the main goal identified in Section 1.3: the design of an FPGA-oriented architecture for baseband processing suitable for multi-waveform coexistence and dynamic spectrum aggregation. The proposed architecture is flexible, scalable, cost-effective, forward-compatible and fulfills performance and reactivity requirements proposed for the next generation of wireless communications. On the way to his contribution, three other are worth mentioning. The first one is an extensive evaluation and assessment of the trade-offs between a DPR-based and a static multi-mode design for a reconfigurable baseband modulator. This contribution fills a research gap previously identified and supports the application of DPR to the design of reconfigurable baseband processing architectures. The second contribution is the first published FPGA-based implementation of a Frequency Spreading FBMC baseband modulator, complemented with a resource utilization comparison with Polyphase Network FBMC designs. Studying and comparing

several architectures and hardware designs for baseband structures is important, because it impacts the cost and energy consumption of future wireless devices. The third contribution is the design of a parallel-pipelined OFDM baseband modulator architecture. This performance of this design is scalable in order to be compatible with different 5G numerologies in a power-efficient way.

1.4.1 Published Work

The work described in this dissertation resulted in the following international publications (by chronological order):

1. Ferreira M.L. and Ferreira J. C., *Reconfigurable NC-OFDM Processor for 5G Communications*, 2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing, Porto, Portugal, 2015, pp. 199-204.

2. Ferreira M.L., Barahimi A. and Ferreira J. C. (2016) *Reconfigurable FPGA-Based FFT Processor for Cognitive Radio Applications*. In Proceedings of the 12th International Symposium on Applied Reconfigurable Computing - Volume 9625, Vanderlei Bonato, Christos Bouganis, and Marek Gorgon (Eds.), Vol. 9625. Springer-Verlag, pp. 223-232.

3. Rodrigues, P; Sinogas, P; Cunha, S; Taing, S; Elsner, J; Uhlenbrock, M; Silva, P; Pessoa, L; Ferreira M.L.; Ferreira, JC; Watts, S: *Simulation and implementation of cognitive radio algorithms for satellite communications*. 67th International Astronautical Congress (IAC 2016)

4. Ferreira M.L., Barahimi A. and Ferreira J. C., *Dynamically reconfigurable FFT processor for flexible OFDM baseband processing*, 2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS), Istanbul, Turkey, 2016.
(*Best Paper Award*)

5. Ferreira M.L., Barahimi A. and Ferreira J. C., *Dynamically reconfigurable LTE-compliant OFDM modulator for downlink transmission*, 2016 Conference on Design of Circuits and Integrated Systems (DCIS), Granada, Spain, 2016.
(*Best Paper Award*)

6. Janßen B., Korkmaz F., Derya H., Hübner M., Ferreira M.L. and Ferreira J. C., *Towards a type 0 hypervisor for dynamic reconfigurable systems*, 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 2017.

7. Carvalho M., Ferreira M.L. and Ferreira J.C., *FPGA-based implementation of a frequency spreading FBMC-OQAM baseband modulator*, 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Batumi, Georgia, 2017, pp. 174-177.

8. Ferreira M.L., Ferreira J.C., Hübner M. (2018) *A Parallel-Pipelined OFDM Baseband Modulator with Dynamic Frequency Scaling for 5G Systems*. In: Voros N., Huebner M., Keramidas

G., Goehringer D., Antonopoulos C., Diniz P. (eds) Applied Reconfigurable Computing. Architectures, Tools, and Applications. ARC 2018, Santorini, Greece, 2018. Lecture Notes in Computer Science, Vol. 10824. Springer, Cham.

9. Ferreira M.L. and Ferreira J.C., *Flexible and Dynamically Reconfigurable FPGA-Based FS-FBMC Baseband Modulator*, 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 2018.

10. Ferreira M.L. and Ferreira J.C., *An FPGA-Oriented Baseband Modulator Architecture for 4G/5G Communication Scenarios*, Electronics Vol.8, Issue 1, art. 2, Jan. 2019.

11. Ferreira M.L. and Ferreira J.C., *A Dynamically Reconfigurable Dual-Waveform Baseband Modulator for Flexible Wireless Communications*, Journal of Signal Processing Systems, Springer.
(accepted for publication)

1.5 Dissertation Structure

This applied research work is founded upon accepted and recognized knowledge produced in the field of Telecommunications, Digital Systems and Embedded Systems Design. Starting from an exhaustive review of the work already developed in these scientific areas, requirements for the target system were defined. The strategy to develop the research work is based on *experiments* and *active research*. Along the research path, prototypes of the target system were developed and subjected to experiments, in order to verify if the requirements initially defined were met. The functional correctness verification of the implemented prototypes was verified by checking the simulation results against the values produced by functional equivalent MATLAB scripts.

This dissertation consists of five further chapters. Chapter 2 contains a bibliographic revision, presenting the fundamental background to produce this PhD work, as well as the state of the art of DPR application to the design of baseband processing architectures. The background knowledge is related to datapath structure and operations for OFDM modulation/demodulation, FBMC and UFMC modulation. The numerologies considered for each waveforms are also presented and discussed. All the presented datapaths require FFT/IFFT modules. For this reason, FFT algorithms and architectures are reviewed and compared. Additionally, the factors that influence the DPR overhead, as well as techniques to mitigate them are presented. All FPGA technical aspects follow the terminology used by Xilinx tools and devices. The state of the art review of DPR-based architectures for baseband processing intends to provide a landscape of the field and to identify its strengths and weaknesses.

Due to their importance and considerable computational complexity, hardware implementations of FFT/IFFT blocks were studied and designed first - Chapter 3. This chapter presents implementations for the two main types of FFT architectures - *pipelined* and *memory-based* - along with results for resource utilization and power consumption. The benefits of adopting FFT pruning algorithms

are also assessed. These algorithms attempt to reduce the FFT/IFFT complexity and execution time in scenarios characterized by considerable data sparsity, by removing operations whose input values are zero and, thus, do not need to be performed. The outcomes from Chapter 3 are the basis for the hardware implementations of baseband processing datapaths presented in Chapter 4. The datapaths considered here are those studied in Chapter 2. These implementations are the starting point to identify potential reconfiguration opportunities and system partitioning strategies explored in this work. In particular, section 4.2 draws heavily on the author's previously published work (Carvalho et al., 2017).

Chapter 5 is devoted to dynamically reconfigurable architectures for baseband processing and divided in three sections. The first section comprises three studies on the application of DPR to dynamically reconfigurable baseband modulators. The first study considers a reconfigurable OFDM modulator and compares two different strategies to define and dimension reconfigurable partitions. Then, a second study evaluates tho different granularity levels for reconfiguration on a dynamically reconfigurable Frequency Spreading FBMC modulator. The third DPR study compares a DPR-based OFDM modulator with a functionally equivalent static multi-mode design in terms of processing throughput, resource utilization, functional density and power consumption. Moreover, the DPR energy overhead is evaluated, through real-time power measurements. This experiment showed that DPR energy and latency overheads are acceptable within the considered application domain. Moreover, results also highlighted the enhanced flexibility, resource efficiency and forward-compatibility of the DPR approach. The second section of Chapter 5 explores a parallel-pipelined architecture for an OFDM modulator, in order to achieve high processing throughputs at low clock frequencies. This is combined with run-time clock frequency adaptation though *dynamic frequency scaling* (DFS). The design was validated for 5G numerologies and the results show the power-efficiency of the proposed approach.

The last section of of Chapter 5 builds upon the previous ones and produces an architecture comprising three independent baseband processing blocks that can be used to modulate different component carriers in non-contiguous carrier aggregation scenarios. Supporting OFDM, FBMC and UFMC, this architecture is also suitable for multi-waveform coexistence scenarios. According to the communication demands, the mode of operation of each baseband processing block is dynamically adapted through DPR and the system clock frequency is adapted through DFS. The idea is to combine the flexibility, scalability, resource-efficiency and forward-compatibility of DPR with the power-efficiency of DFS. This experiment clearly shows how DPR-based hardware virtualization can enable a smaller and cost-effective device to be used in the implementation of a complex and resource demanding system. The contents on Chapter 5 draw heavily on the author's previously published work (Ferreira et al., 2016; Ferreira and Ferreira, 2018; Ferreira et al., 2018; Lopes Ferreira and Canas Ferreira, 2019).

Finally, Chapter 6 contains the main conclusions of this dissertation, as well as future work and research directions.

Chapter 2

Background and State of the Art

This chapter presents the background knowledge on which this dissertation is based. First, multi-carrier modulation waveforms, in particular OFDM, UFMC and FBMC, are discussed. Due to its relevance in this domain, special attention is given to FFT algorithms and architectures. Then, topics on run-time reconfigurability on FPGA are presented. Finally, the state-of-the-art of multi-mode baseband processors employing dynamically reconfigurable techniques is discussed.

2.1 Multicarrier Modulation Waveforms

Multicarrier modulation (MCM) is a Frequency Division Multiplexing (FDM) method widely adopted for high data rate transmission. It divides a wideband data stream into several streams that are simultaneously used to transmit data at different subcarrier frequencies (Bingham, 1990). Considering S multicarrier symbols of period T_{sym} with N subcarriers each, the generic expression to represent an MCM waveform in the time domain is:

$$x(t) = \sum_{s=0}^{S-1} \sum_{i=0}^{N-1} d_{s,i} \cdot p(t - sT_{sym}) \cdot e^{j2\pi i f_i (t - sT_{sym})}, \tag{2.1}$$

where $d_{s,i}$ is the QAM/PSK modulated symbol transmitted at the i^{th} subcarrier of the s^{th} multicarrier symbol; $p(t)$ is a function for subcarrier pulse shaping in the frequency domain; and the exponential term is the frequency shifter for the i^{th} subcarrier. A time domain bandpass filter *b(t)* can be also used to improve the OOB performance of MCM systems. Figure 2.1 shows the general model for MCM communications. The subcarriers are individually modulated and then summed before transmission. On the receiver side, the subcarriers are separated before demodulation. In this work, the MCM waveforms considered are OFDM, UFMC and FBMC. These waveforms mainly differ from each other regarding the pulse shaping function *p(t)* and the use of the time-domain filter *b(t)*. In this work, modulation and demodulation are considered for OFDM, whereas only modulation is studied for UFMC and FBMC.

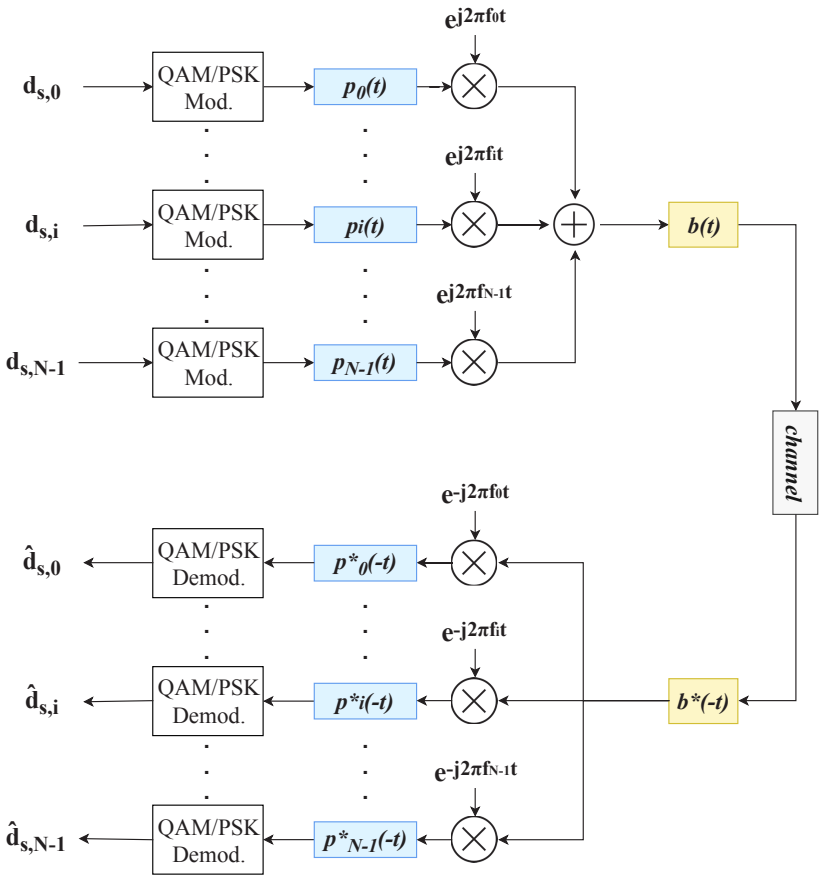


Figure 2.1: General model for Multicarrier Modulation communications; $d_{s,i}$ is a QAM/PSK modulated symbol transmitted at the i^{th} subcarrier of the s^{th} multicarrier symbol; $p(t)$: pulse shaping function; *b(t)*: time domain bandpass filter.

2.1.1 OFDM

OFDM is the main reference among MCM waveforms in wired and wireless communications. It is used in a wide range of standards such as DSL, DVB-T, DVB-C, WiFi (IEEE 802.11), WiMAX (IEEE 802.16) and 3GPP LTE. In OFDM systems, the pulse shaping function *p(t)* is a *sinc* pulse. However, it is applied in the time domain as a rectangular pulse window with duration T_{sym}. The frequency spacing (Δf) between adjacent subcarriers is equal to the inverse of the symbol period: $\Delta f = 1/T_{sym}$. This ensures orthogonality between subcarriers, i.e. the frequency spectrum of each subcarrier is null at the center frequency of the other subcarriers (Hwang et al., 2009). No time domain filter *b(t)* is applied. An individual OFDM symbol (e.g. for $s = 0$) is represented by the expression:

$$x(t) = \sum_{i=0}^{N-1} d_{s,i}.p(t).e^{j2\pi i\Delta f t} \tag{2.2}$$

Sampling it at instants $t = n\frac{T_{sym}}{N}$, with n integer, we have:

$$\begin{aligned} x[n] &= \sum_{i=0}^{N-1} d_{s,i}.e^{\frac{j2\pi iT_{sym}n}{T_{sym}N}} \\ &= \sum_{i=0}^{N-1} d_{s,i}.e^{\frac{j2\pi in}{N}} \end{aligned} \tag{2.3}$$

This expression resembles the N-points Inverse Discrete Fourier Transform (IDFT) operation, which can be efficiently implemented using Inverse Fast Fourier Transform (IFFT) algorithms. Analogously, the Fast Fourier Transform (FFT) algorithm is used for OFDM demodulation. In an OFDM symbol, not all N subcarriers are used to transmit data. Apart from *data* subcarriers, there are *pilot* subcarriers and *null* subcarriers. Pilot subcarriers, whose value and location are defined for each standard, are used for channel estimation and equalization at the receiver side. Null subcarriers normally serve as guard bands at the edges of the spectrum band in use. The guard band insertion reduces spectral efficiency, as the amount of spectrum resources used for actual transmission is lower, but alleviates OOB interference. Prior to IFFT, the subcarriers are mapped to the N IFFT points according to their type - *subcarrier mapping*.

In multi-path fading channels, *inter-symbol interference* (ISI) between adjacent OFDM symbols affects the correct symbol sampling at the receiver, causing the degradation of the system error rate. To counter this effect, OFDM systems typically prefix each multicarrier symbol by part of the symbol's end - *cyclic prefix* (CP) insertion. To effectively mitigate inter-symbol interference, the CP length (L_{CP}) must be greater than the channel delay spread, which is a measure of the multipath reflections on the channel.

OFDM does not provide genuinely band-limited signals due to the *sinc* pulse side lobes. To improve OFDM spectral containment, a non-rectangular time-domain window is applied to the CP-extended symbols. This window is characterized by smooth transitions between adjacent multicarrier symbols. Raised cosine windows are often used to produce windowed OFDM symbols. In practical applications, such as LTE downlink transmission, windowed OFDM symbols adjacent

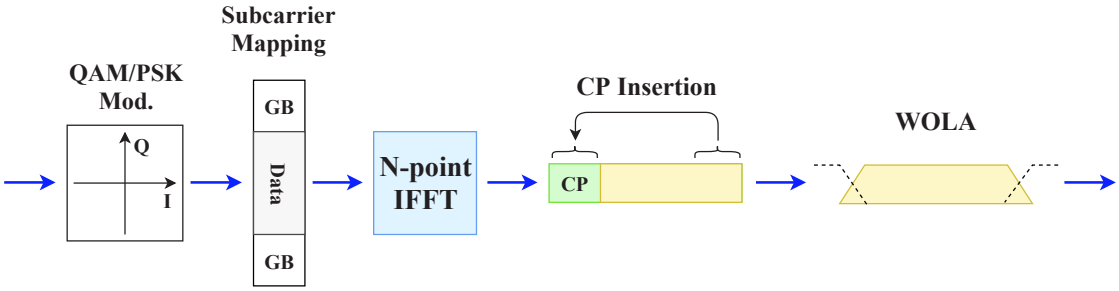


Figure 2.2: OFDM baseband modulation. *GB*: zero-valued guard bands.

to each other are overlapped in the edge transition region - *weighted overlap-and-add* (WOLA). The general structure for OFDM modulation is depicted in Figure 2.2.

A baseline OFDM waveform is synthesized with the IFFT operation. For sake of simplicity, in this work, the term OFDM refers to an OFDM waveform that employs CP insertion and WOLA. In the literature, it is common to differentiate between CP-OFDM (baseline OFDM with CP insertion) and Windowed-OFDM (Zaidi et al., 2018) or WOLA-OFDM (Lien et al., 2017) (baseline OFDM with CP insertion and WOLA).

From the OFDM waveform synthesis just described, two operations are obviously required for waveform analysis at the receiver side: CP removal and FFT. However, the high OFDM sensitivity to synchronization errors requires additional operations for baseband demodulation. The two main uncertainties at the receiver side are the symbol time and carrier frequency offsets. Symbol time offset (STO) is due to the different time references used at the transmitter and receiver, which complicates the detection of the symbol start at the receiver. In turn, the carrier frequency offset (CFO) arises from differences in the local oscillators at the transmitter and receiver. Its main effect is the loss of orthogonality between OFDM subcarriers. Before CP removal, coarse STO and CFO estimation and correction are performed. The study and review of synchronization methods for OFDM systems is out of the scope of this work.

After coarse synchronization, the CP is removed and the FFT converts the received symbol from the time to the frequency domain. Fine STO mechanisms can be applied after the FFT to complement the initial coarse STO estimation (Chang, 2008). The distortions introduced by multi-path fading channels cannot be completely overcome with synchronization methods. Consequently, *channel estimation* and *equalization* are also performed. The estimation of a non-flat channel can be achieved through pilot-assisted techniques (Hwang et al., 2009). Usually, the value and location of pilot subcarriers is determined by the communication standard and is known at the receiver. Based on the received and expected pilot values, it is possible to estimate the channel frequency response at the pilot frequencies. Channel equalization can be obtained through pilot interpolation, in order to determine the channel response for non-pilot frequencies. Again, the review and study of fine STO, channel estimation and equalization is out of the scope of this work. Figure 2.3 shows the general datapath structure for an OFDM baseband demodulator.

A spectral agile variant of OFDM - Non-Contiguous OFDM (NC-OFDM) - has been proposed

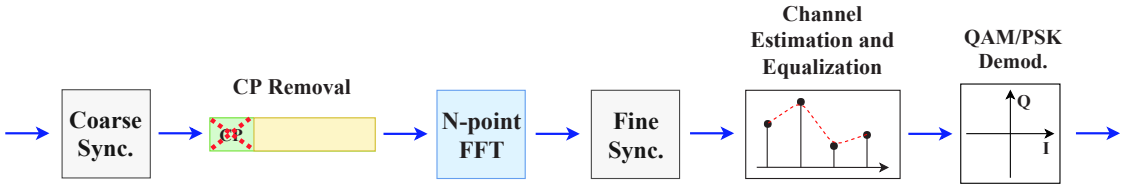


Figure 2.3: OFDM baseband demodulation.

for the flexible utilization of fragmented portions of the spectrum Bogucka et al. (2015). To avoid interferences with licensed user transmissions, NC-OFDM deactivates certain blocks of sub-carriers by setting them to zero. There are considerable challenges related with the application of NC-OFDM in real systems, especially related to synchronization and channel estimation. In dynamic spectrum access environments, the available sub-carriers for transmission are not always the same and fixing frequency locations for pilot carriers could result in interferences with licensed users. Solutions for NC-OFDM synchronization (Dutta et al., 2010b) and channel estimation (Chen and Chu, 2012) have been proposed, considering the challenges imposed by dynamic carrier allocation. In scenarios of heavy licensed user occupation, the FFT/IFFT inputs will have a considerable amount of zeros. This data sparsity can be used to reduce the computational complexity of the FFT/IFFT operation through FFT *pruning* techniques (Alves et al., 2000).

2.1.2 UFMC

UFMC, also known as Universal Filtered OFDM (UF-OFDM), is an OFDM-based waveform that attempts to reduce OOB emissions through time-domain filtering. Recalling the general MCM structure from Figure 2.1, UFMC uses the same pulse shaping function p(t) as OFDM, but then applies a bandpass time-domain filter *b(t)* on sets of subcarriers - *subbands* or *physical resource blocks* (PRBs). The N subcarriers per symbol are divided into B PRBs of $\frac{N}{B}$ subcarriers each. Usually, only part of the PRBs is used for transmission - *active PRBs*. For each active PRB, IFFT and bandpass L-order FIR filtering are performed. Instead of the CP, a zero-valued guard interval with length L is inserted after the IFFT. Frequency-shifted versions of the FIR filter are applied to all active PRBs and, finally, the filtered subbands are superimposed to form an UFMC multicarrier symbol for transmission. Chebyshev filters are normally used for bandpass filtering in UFMC (Wang et al., 2014; Jafri et al., 2018; Nadal et al., 2017).

The classic UFMC modulation scheme (Vakilian et al., 2013) considers an N-point IFFT and FIR filters with complex coefficients for each active subband. To reduce this increased complexity, Knopp et al. (Knopp et al., 2016) combine a smaller N'-point IFFT with $\frac{N}{N'}$ upsampling. Moreover, the same real-coefficient FIR filter is used in all subbands, followed by frequency shifters implemented with multiplications by a complex exponential. Figure 2.4 illustrates the datapath structure for the UFMC modulation considered in this work.

As opposed to the situation with OFDM, there are few hardware implementations for UFMC baseband modulators. Jafri et al. (Jafri et al., 2018) follow the algorithm proposed in (Knopp et al.,

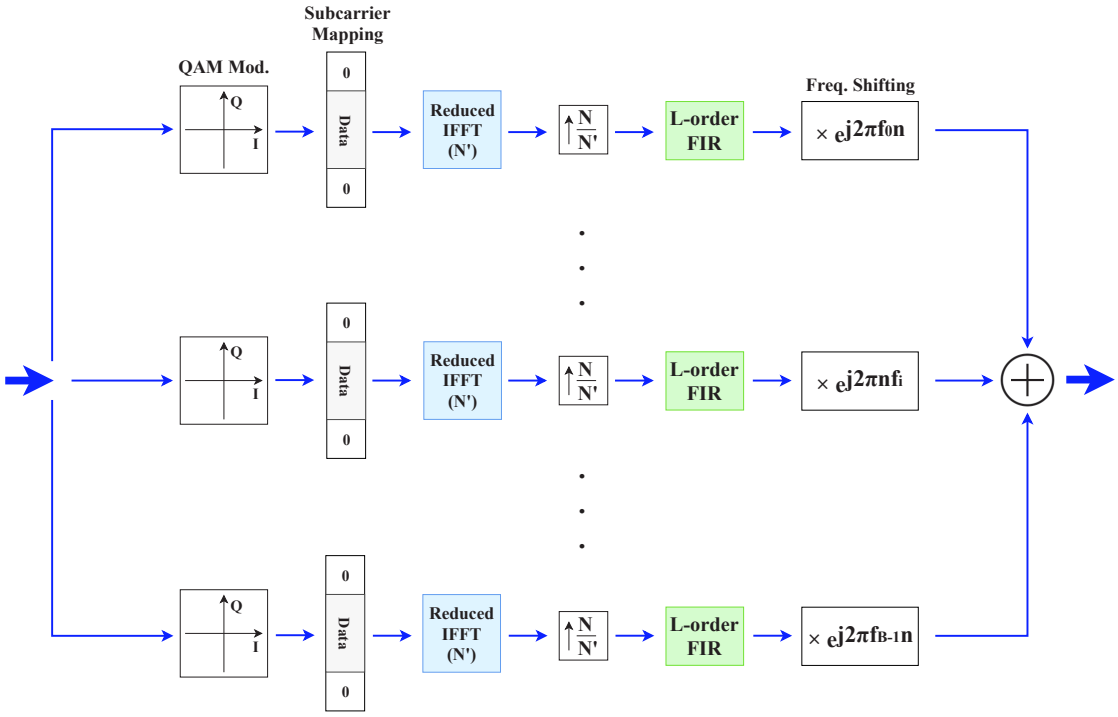


Figure 2.4: UFMC baseband modulation.

2016) and present an FPGA-based UFMC modulator for LTE with 10 MHz channel bandwidth. Each UFMC subband is not processed in parallel, but using a *ping-pong* buffering/memory strategy, which requires clock frequencies above 300 MHz to achieve acceptable processing latencies. In turn, Medjkouh, et al. (Medjkouh et al., 2017) present an FPGA-based implementation for a baseband modulator that exploits the separation between subband and subcarrier processing and the decomposition of the UFMC symbol into prefix, core and suffix parts. With this technique, the authors claim a significant complexity reduction compared to a baseline UFMC implementation, which can be very important when the number of allocated subbands for increases.

2.1.3 FBMC

While the WOLA operation improves OFDM OOB performance through time-domain windowing, FBMC achieves a better spectral confinement by applying an array of subcarrier-wise prototype filters - *filter bank* - in the frequency domain. Designing prototype filters with long decaying tails provides robustness against inter-symbol interference and avoids the use of CP or guard intervals (Zaidi et al., 2018). This reduces transmission redundancy and increases spectral efficiency. The prototype filter design uses frequency sampling techniques (McCreary, 1972), leading to the spanning of the filter over several subcarriers. This breaks the orthogonality between subcarriers and causes the overlapping of multicarrier symbols in the time domain (Bellanger, 2012). To compensate for the loss of orthogonality, FBMC is often combined with *Offset Quadrature Amplitude Modulation* (OQAM) to keep real-part orthogonality between symbols (Schellmann et al., 2014).

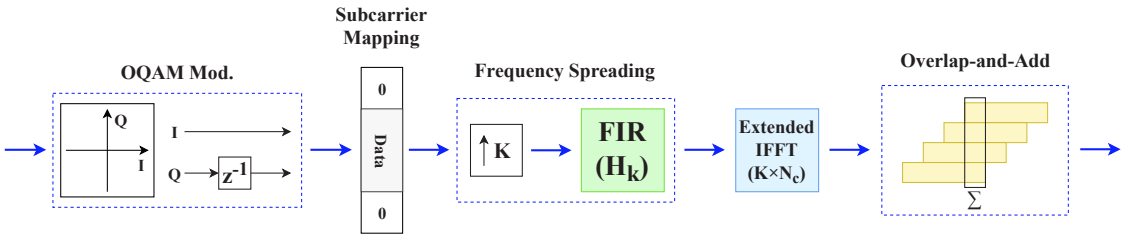


Figure 2.5: Frequency Spreading FBMC-OQAM baseband modulation.

Due to the OQAM transmission of the real and imaginary components, the FBMC modulator has to run at twice the frequency of an equivalent OFDM modulator to achieve the same data rate. The waveform design combining FBMC and OQAM is referred in literature as FBMC-OQAM. In this work, we only studied FBMC-OQAM baseband modulation and, therefore, all mentions to FBMC refer to FBMC-OQAM.

The synthesis filter bank depends on two parameters - the overlapping factor (K) and the number of subcarriers per symbol (N_c) - and can be implemented using two strategies: Frequency Spreading FBMC (FS-FBMC) (Bellanger, 2012) and Polyphase Network FBMC (PPN-FBMC) (Siohan et al., 2002). FS-FBMC follows the general multicarrier scheme from Figure 2.1, with subcarrier filtering performed in the frequency domain. The frequency spreading operation consists of upsampling the OQAM symbols by K and FIR filtering. As a consequence, the IFFT size is extended to $K \times N_c$. Finally, FS-FBMC with OQAM requires the *overlap-and-add* of consecutive IFFT output stream blocks delayed by $N_c/2$ samples (Doré et al.). In contrast, the PPN-FBMC approach keeps the N_c-points IFFT and performs the filtering operation through a time-domain Polyphase Network (PPN). The PPN is an array of phase-shifted versions of the prototype filter placed in series with the IFFT block. PPN-FBMC has a lower computational complexity than FS-FBMC, due to the smaller IFFT size. However, the FS-FBMC scheme simplifies the FBMC concept, making it easier to compare with other MCM waveforms. Moreover, for a fixed IFFT size, FS-FBMC offers the flexibility of changing N_c by keeping the IFFT size and simply varying K (Bellanger, 2012).

In this work, the frequency spreading approach was adopted to implement an FBMC baseband modulator. The general datapath structure of the FS-FBMC-OQAM baseband modulator is shown in Figure 2.5.

The *PHYDYAS* project proposed a family of prototype filters widely used in FBMC systems (Bellanger et al., 2010). It is based on the Nyquist criterion and its impulse response is given by:

$$p(t) = 1 + 2\sum_{k=0}^{K-1} H_k \cdot \cos(2\pi \frac{kt}{KT_{sym}}) \tag{2.4}$$

, where K is the overlapping factor and T_{sym} is the multicarrier symbol period. The number of non-zero filter coefficients (H_k) is $2 \times K - 1$ and their values are shown in Table 2.1, for $K = 2, 3, 4$. OFDM can be viewed as a particular FBMC case where $K = 1$, i.e. there are no overlapped symbols in the time domain and the prototype filter has only one non-zero coefficient of value 1.

Although the first developments related with FBMC date back to the 1960s, only recently

Table 2.1: Frequency domain prototype filter coefficients (Bellanger et al., 2010).

K	H_0	$H_1 = H_{-1}$	$H_2 = H_{-2}$	$H_3 = H_{-3}$
2	1	$\sqrt{2}/2$	-	-
3	1	0.911438	0.411438	-
4	1	0.971960	$\sqrt{2}/2$	0.235147

has it been considered as a promising multicarrier technique. In the context of wireless communications, research works on FBMC mainly focus on waveform simulation (Viholainen et al., 2009; Weitkemper et al., 2016; Won et al., 2017) or architecture proposals (Schaich, 2010; Varga and Kollár, 2013). Few hardware implementations have been implemented so far. Regarding FPGA-oriented implementations for FBMC-OQAM baseband modulators, the few published works follow a PPN approach to implement the synthesis filter bank. Berg et al. (Berg et al., 2014) propose a flexible FBMC-based transceiver for opportunistic spectrum access in TV white spaces. Although FS-FBMC is used on the receiver side, the modulator on the transmitter side follows a PPN approach. A complexity evaluation and comparison with an OFDM transceiver is done and the FBMC system showed a hardware resource overhead of approximately 30%. The system is implemented on a Xilinx Kintex-7 FPGA.

With the purpose of reducing computational complexity, a PPN-FBMC-OQAM pipelined architecture is proposed in (Nadal et al., 2016). IFFT pruning algorithms are used to halve the IFFT size. However, a pre-processing unit is required. The hardware platform chosen is an Xilinx Zynq device and Post-Synthesis results for resource utilization, power consumption and latency are presented. Two implementations are presented for different PPN variants. The proposed architecture reduces the complexity gap to OFDM. The paper claims a maximum clock frequency of 220 MHz, corresponding to the propagation delay of a DSP primitive.

Waveform comparison Despite the adoption of OFDM in the 3GPP 5G NR PHY Release 15 (TS, 2018), it is unlikely that a single waveform will satisfy all the requirements imposed by future wireless communications. Thus, waveform coexistence is a probable near-future scenario (Kaltenberger et al., 2015; Jue, 2017). From the three waveforms presented, OFDM is a mature technique that combines a relatively lower computational complexity with a satisfactory overall performance. It is compatible with previous generation communications and its flexibility enables application in MIMO systems. In turn, with the careful filtering of each individual subcarrier, FBMC avoids the use of CP or guard intervals and shows better spectral efficiency and confinement than OFDM-based waveforms. This makes FBMC well-suited for fragmented spectrum use in Cognitive Radio scenarios (Farhang-Boroujeny, 2011). UFMC finds a convenient application in sporadic and short burst transmission communications, due to subband filtering (Schaich et al., 2014). A common operation in all baseband processing datapaths previously presented is the an DFT/IDFT operation. For this reason, the next section focus on algorithms and architectures for efficient DFT/IDFT implementation.

2.2 Fast Fourier Transform

In the 19th century, the French mathematician Jean-Baptiste Joseph Fourier showed that a periodic function can be decomposed into an infinite sum of trigonometric functions - *Fourier representation*. As the trigonometric functions are periodic signals with well defined frequencies, the Fourier representation allows us to analyze a signal in terms of the frequencies which compose it - *frequency spectrum*. The mathematical operation which converts a signal from the time domain to the frequency domain was then named the *Fourier Transform*. The original formulation of the Fourier Transform considers continuous signals of infinite duration and bandwidth. However, in DSP practical implementations, it is more convenient to use a version of the Fourier Transform sampled in the time and frequency domains - *Discrete Fourier Transform* (DFT). Apart from spectral analysis of signals, the DFT finds numerous applications in the domains of image/audio signal processing or digital communications. An N-point DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \qquad k = 0, 1, \ldots, N-1 \tag{2.5}$$

where N is the DFT size, n the time index and k the frequency index. The terms W_N^{nk} are commonly known as *twiddle factors* and represent the N^{th} roots of unity: $W_N^{nk} = e^{-\frac{j2\pi}{N}nk}$. The multiplication by twiddle factors is also called *rotation*. The DFT counterpart - *Inverse Discrete Fourier Transform* (IDFT) - recovers the original signal from the frequency domain to the time domain. Its expression is:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}, \qquad n = 0, 1, \ldots, N-1 \tag{2.6}$$

The computational complexity of the DFT is O(N^2), which becomes critical for higher values of N. To face this problem, several algorithms to efficiently compute the DFT were developed, generally known as *Fast Fourier Transform* (FFT) algorithms. The base for most of the FFT algorithms is a *divide and conquer* strategy, where the original problem to solve - the DFT - is divided in smaller, easier problems. More specifically, the original DFT is broken down into several smaller DFTs, in a way that the sum of the computational effort of each smaller DFT is lower than the computational effort of the original DFT. Good (Good, 1971) and Thomas (Thomas, 1963) proposed a *divide-and-conquer* FFT algorithm where the twiddle factor multiplications are eliminated. This is achieved through the decomposition of the original DFT into DFTs whose sizes are co-prime. The price to pay for the removal of twiddle factor multiplications is a complicated index mapping of the input data. Another approach is the DFT conversion into a cyclic convolution - Rader's permutation - followed by the application of Winograd's cyclic convolution algorithm (Winograd, 1978). Again, the complex index mapping associated with this algorithm makes its hardware implementation complicated.

The *Cooley-Tukey algorithm* (Cooley and Tukey, 1965) is the most widely used FFT algorithm. It offers a systematic procedure to compute the FFT for any factorization of N, with a reasonable computational complexity - O(Nlog(N)) - and efficient mapping to hardware implementations. N is

decomposed in two factors - $N = N_1 \times N_2$ - and a two-dimension index mapping is performed in the time (n) and frequency (k) domains:

$$n = N_2 n_1 + n_2 \begin{cases} 0 \leq n_1 \leq N_1 - 1 \\ 0 \leq n_2 \leq N_2 - 1 \end{cases} \tag{2.7}$$

$$k = k_1 + N_1 k_2 \begin{cases} 0 \leq n_1 \leq N_1 - 1 \\ 0 \leq n_2 \leq N_2 - 1 \end{cases} \tag{2.8}$$

This approach explores twiddle factor symmetries and periodicity:

$$\begin{gathered} W_N^{N_1} = W_{N_2}, \\ W_N^{N_2} = W_{N_1}, \\ W_N^k = W_N^{k+N} = -W_N^{k+N/2}, \end{gathered}$$

transforming 2.5 into:

$$X[k_1, k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left(W_N^{n_2 k_1} \sum_{n_1=0}^{N_1-1} x[n_1, n_2] W_{N_1}^{n_1 k_1} \right). \tag{2.9}$$

Defining the expression inside parentheses as $\bar{x}[n_2, k_1]$ (Meyer-Baese, 2007) we have:

$$X[k_1, k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \bar{x}[n_2, k_1] \tag{2.10}$$

In fact, this expression is the computation of N_2 DFTs of size N_1, followed by the computation of N_1 DFTs of size N_2. Despite allowing any factorization of N, most Cooley-Tukey implementations consider values of N such that $N = r^s$. This sub-class of algorithm is commonly referred as *Radix-r FFT algorithms*. The method consists of recursively decomposing the original N-point FFT into smaller FFTs, until a size less or equal to r is achieved. The number of recursive stages is equal to $log_r(N)$. In a Radix-r FFT algorithm, the basic computational block that performs an r-point FFT is designated as *butterfly*. Radix-2 FFT algorithms are extremely popular due to the reduced butterfly complexity. Recalling 2.5, a 2-point DFT is computed as:

$$\begin{aligned} \begin{bmatrix} X[0] \\ X[1] \end{bmatrix} &= \begin{bmatrix} e^0 & e^0 \\ e^0 & e^{-j\pi} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} \end{aligned} \tag{2.11}$$

From this expression, it is clear that, the Radix-2 (2.11) butterfly involves trivial twiddle factor multiplications and can be implemented using only adders and/or subtractors. A Radix-2 FFT algorithm for $N = 8$ is exemplified in Figure 2.6. As a result of the $8 = 2^3$ factorization, $log_2(8) = 3$ stages can be clearly identified. In this example, the input sequence appears in natural order

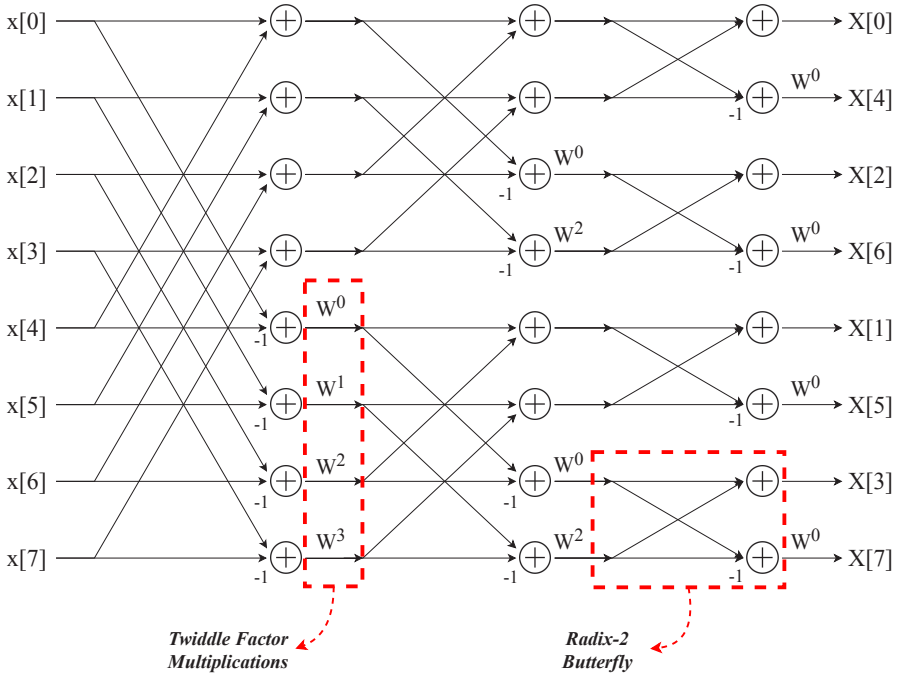


Figure 2.6: Signal flow graph for a 8-points radix-2 DIF FFT.

and the output sequence in bit-reversed order - *Decimation-in-Frequency* (DIF) decomposition. Alternatively, the decomposition can be done in the time domain, so that the input sequence appears in bit-reversed order, while the output appears in natural order - *Decimation-in-Time* (DIT) decomposition. Depending on the decomposition method employed, sample reordering may be required on the input or output sequence in order to present the FFT results in natural order.

Radix-r algorithms with higher radixes have fewer FFT stages and butterflies, but introduce non-trivial twiddle factor multiplications on the butterflies. The Radix-2^2 FFT algorithm (He and Torkelson, 1996) has the same multiplicative complexity as the Radix-4, but keeps the simplicity of Radix-2 butterflies. This is achieved by exploring the fact that multiplications by $\pm j$ involve the swapping of real and imaginary parts and sign inversion. The Radix-2^2 algorithm can be generalized for 2^n radixes. However, for non power-of-two radixes, non-trivial twiddle factor multiplications are unavoidable. It is also possible to mix several radixes in the same algorithm - *Mixed-Radix FFT* - in order to achieve FFT sizes that are not powers-of-two. Figure 2.7 presents the signal flow graphic for a Mixed-Radix algorithm to compute a 12-point FFT, using Radix-4 and Radix-3 butterflies.

A straightforward approach to implement an FFT core is the *direct implementation* of the algorithm flow graph into hardware. This way, there would be a dedicated hardware block to implement every processing element (*butterflies* and/or *complex multipliers*) in the flow graph. This approach is easy to implement and provides high data throughput. However, it requires a high amount of resources and circuit area. The two main categories of FFT architectures are *Memory-based* and *Pipelined*. The choice between them depends on design specifications such as hardware resource utilization, latency, data throughput or power consumption.

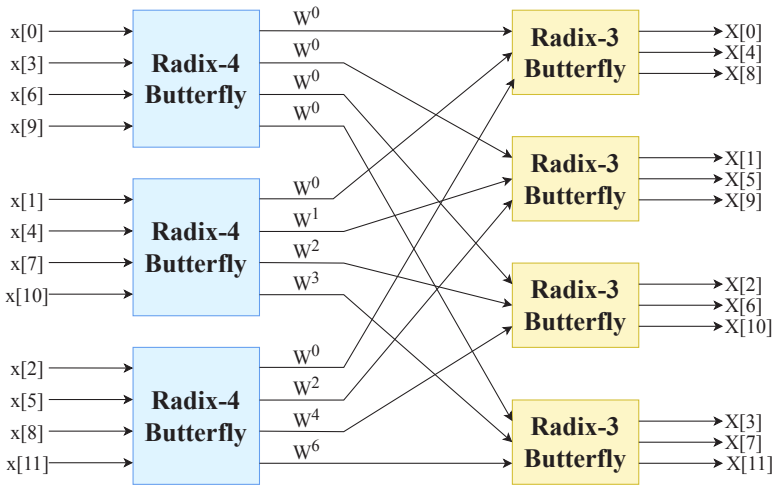


Figure 2.7: Signal flow graph for a 12-point Radix-2 DIF FFT with factorization $N = N_1 \times N_2 = 4 \times 3$. The time-frequency index mapping follows 2.7 and 2.8 (Meyer-Baese, 2007).

In memory-based FFT architectures (Tsai et al., 2006; Hidalgo et al., 1999; Xiao et al., 2008b), also known as *in-place* architectures, all the additions and multiplications involved in the FFT computation are carried out by few processing elements - usually, no more than two butterflies and two complex multipliers. To allow this intensive reuse of processing elements, memory banks are required to store input values and intermediate calculations. Figure 2.8 shows a basic memory-based architecture. Its operation can be divided in two phases: *load input/unload results* and *process transform*. During the *load input/unload results* phase, the control engine issues read/write operations on the memory banks to fill them with the incoming data samples. At the same time, the results from previous transform processing are sent to the output. The *process transform* phase corresponds to the execution of the processing steps of each FFT processing stage. An immediate advantage of this architecture is the low amount of resources and circuit area required. On the other hand, this mode of operation is iterative and does not explore hardware's potential for parallel execution. This leads to lower data throughput and high processing latency, mainly for large FFT sizes. The main concern in memory-based architectures has to do with the memory accesses and addressing schemes, which can become quite challenging for higher radixes.

In CR and DSA scenarios, the available sub-carriers can be sparsely distributed along the spectrum and a high number of FFT/IFFT inputs will be zero. FFT *pruning* algorithms attempt to reduce the FFT/IFFT complexity by: "*removing operations on input values which are zero, and on output values which are not required*" (Alves et al., 2000). Although pruning algorithms can potentially reduce the FFT computational complexity and improve the system's energy efficiency, they require the pre-processing of the incoming data in order to generate a pruning matrix with information about the FFT operations to be pruned. The iterative nature of memory-based architectures allows this data pre-processing and the exploitation of *pruning* algorithms. As the *load input/unload results* and *process transform* phases are not simultaneous, it is possible to mark zero-valued operations and avoid their execution. The work from (Jang et al., 2011) employs an efficient zero flag detection

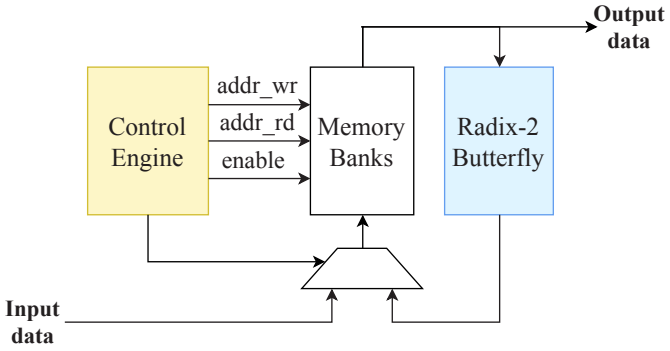


Figure 2.8: Memory-based FFT architectures.

scheme, while (Airoldi et al., 2015) uses a memory to store a pruning matrix that indicates which operations to execute at every FFT stage.

In Pipelined architectures (He and Torkelson, 1996; Garrido et al., 2013; Cortés et al., 2008), all operations at an FFT stage are performed by a single butterfly and complex multiplier. Each pipeline stage also requires some memory in order to correctly synchronize and schedule the operations to execute in each stage. This category of FFT architectures shows a regular structure, requires a reasonable amount of resources and a relatively simple control logic. Compared with memory-based architectures, the pipelined architectures require more resources and circuit area, but allow for higher data throughput and the processing of continuous data streams. In contrast with memory-based architectures, pipeline architectures start the computation as soon as the first input sample is received. This continuous processing nature makes pipelined FFT architectures better suited for real-time applications (He and Torkelson, 1998). The operation control in pipelined architectures is typically done using simple counters.

Garrido et al. (Garrido et al., 2013) consider two main kinds of pipelined architectures: *Feedback* (FB) and *Feedforward* (FF). In the FB architectures, the butterflies have feedback loops (shift registers) to correctly pair the input samples for further processing. The simplest implementation of FB architectures is the *Single-Path Delay Feedback* (SDF) (He and Torkelson, 1996; Cortés et al., 2008) whose structure is presented in Figure 2.9 for the Radix-2 case. This architecture produces only one sample per clock cycle. In high throughput applications, several parallel SDF datapaths can be used together - *Multi-Path Delay Feedback* (MDF) architecture. The throughput improvement in MDF comes at at the cost of replicating hardware resources. Instead of feedback shift registers, FF architectures employ feed-forward buffers and switching elements to store and reorder data between processing stages. This type of architecture is often designated as *Multi-path Delay Commutator* (MDC) (Figure 2.10). MDC architectures produces several output samples in parallel, presenting a higher throughput than SDF architectures. The choice between FB and FF architectures mainly relies on a *throughput vs. resources* trade-off. Table 2.2 presents a comparison between SDF and MDC architectures for Radix-2, Radix-4 and Radix-2^2 algorithms.

All the aspects discussed in this section also hold for the IFFT (2.6). The IFFT computation can be implemented using an FFT core with real/imaginary part swapping on the input/output

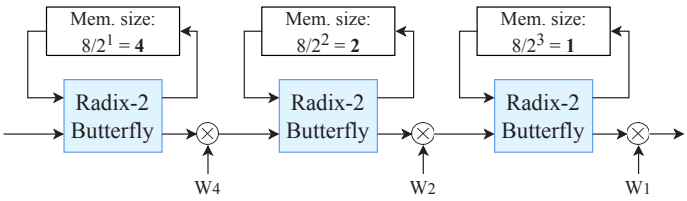


Figure 2.9: Schematic of a Radix-2 SDF pipelined FFT architecture for $N = 8$. The memory size is quantified as the amount of complex samples.

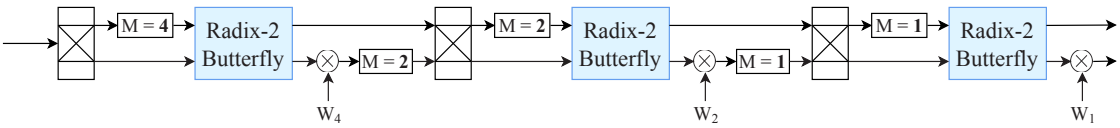


Figure 2.10: Schematic of a Radix-2 MDC pipelined FFT architecture for $N = 8$. The memory size is quantified as the amount of complex samples.

Algorithm	Architecture	# complex multipliers	# adders	Memory requirements
Radix-2	SDF	$2 \times (log_4(N) - 1)$	$4 \times log_4(N)$	$N - 1$
	MDC	$2 \times (log_4(N) - 1)$	$4 \times log_4(N)$	$3 \times N/2 - 2$
Radix-4	SDF	$log_4(N) - 1$	$8 \times log_4(N)$	$N - 1$
	MDC	$3 \times (log_4(N) - 1)$	$8 \times log_4(N)$	$5 \times N/2 - 4$
Radix-2^2	SDF	$log_4(N) - 1$	$4 \times log_4(N)$	$N - 1$
	MDC	$2 \times (log_4(N) - 1)$	$4 \times log_4(N)$	$N - 2$

Table 2.2: Hardware requirements comparison for SDF and MDC architectures, considering Radix-2, Radix-4 and Radix-2^2 algorithms. Expressions extracted from (He and Torkelson, 1996) and (Garrido et al., 2013).

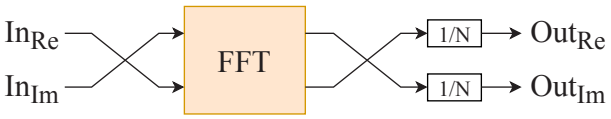


Figure 2.11: Scheme to obtain an IFFT from and FFT core

sequences and a multiplication by $\frac{1}{N}$ (Figure 2.11).

Due to its relevance regarding the baseband processing for multicarrier modulation waveforms, some emphasis was given to FFT/IFFT algorithms and architectures. Moreover, the design of the baseband processors presented in this work started with the hardware implementation of FFT/IFFT cores. The contents of the current section are the basis for the FFT cores presented in Chapter 3.

2.3 FPGA-oriented Multi-mode Baseband Processors

The study and design of reconfigurable baseband processors is often associated of SDR and CR research efforts (Akeela and Dezfouli, 2018). When trying to combine flexibility, scalability, resource-power efficiency and forward-compatibility, several design approaches have been considered. GPPs are highly flexible, easy to program and upgrade with new functionalities. However, their sequential execution model leads to very high power consumption to achieve a throughput compatible with real-time specifications. Performance can be improved by accelerating certain tasks in graphics processing units (GPUs), but real-time requirements can be hampered by the overhead associated with the GPP-GPU interconnection (Li et al., 2014). GPP-based approaches are well suited for complex computation, data storage and connectivity purposes, making them an option for baseband processing in base stations. Digital Signal Processors (DSPs) have the same advantages of GPPs, but yield better throughput performance due to their partially parallel execution model. Depending on the manufacture process, DSPs can be suited for user terminals (optimized for energy) or base stations (optimized for performance). In turn, ASICs can achieve high throughput rates in a power-efficient way. However, they are not forward-compatible and have a very low flexibility and programmability. Consequently, ASICs are particularly suited for massive production user terminals or ultra-high performance base stations with a constrained set of functionalities.

An ideal reconfigurable baseband processor would combine the flexibility and easy programmability of GPPs with the performance of ASICs. Based on this, Tang et al. (Tang et al., 2013) propose a reconfigurable baseband core for multi-mode communication transmission composed of a 65 nm pipelined co-processor controlled and configured by a GPP by pipeline stage activation/deactivation and parameter values setting. Although the architecture may achieve a satisfactory performance for WLAN and UWB standards, the GPP impact in the global power consumption is not evaluated. The authors claim the support for new standards and modes of operation can be implemented in the GPP. However, given the increasing data rates required by future wireless communications (Zaidi et al., 2016), it is unlikely that a software-based GPP approach can satisfy performance requirements.

FPGAs provide a balanced trade-off between the GPP-GPU, DSP and ASIC advantages. Albeit less flexible than GPPs or DSPs, FPGAS still allow for design and run-time reconfigurability. With moderate power consumption and highly parallel execution, FPGAs achieve throughput performances closer to ASICs than to DSPs. This makes them widely used for real-time applications and suitable for flexible and upgradeable base stations. For instance, SDR platforms like WARP (war)

or the Ettus USRP X Series (usr) include FPGA devices as custom high-speed baseband processing engines. The implementation of soft-processors in the FPGA logic fabric also enables the exploration of Hardware-Software co-design methodologies to enhance the programmability of FPGA-based designs. System-on-Chip devices such as the Xilinx Zynq (zyn) or the Intel Cyclone V SoC (int) have also been exploited in SDR/CR scenarios (Shreejith et al., 2015; Drozdenko et al., 2018).

The multitude of OFDM-based wireless standards in 3G/4G raised the interest for implementations supporting multi-mode operation, supporting different FFT/IFFT sizes, cyclic prefix lengths, constellation schemes or number of active subcarriers. In (Dutta et al., 2010a), an FPGA-oriented design for OFDM baseband processing in Software Defined Cognitive Radios is presented. The mode of operation is defined by setting the value of baseband parameters like FFT/IFFT size, CP length or modulation scheme. Baseband processing cores for every considered scenario are always present in the system. Consequently, a large amount of FPGA resources is required to implement the design (around 94% of the resources of a Virtex-IV FPGA). Chacko et al. (Chacko et al., 2014) propose a flexible and latency-insensitive FPGA-based OFDM pipeline for wireless communication systems. The pipeline is composed by several stages responsible for a certain baseband operation. The functional elements are dimensioned for the worst-case scenario and, through parameter values selection, are scaled and modified according to the desired mode of operation. A similar approach is followed in (Zhang and Guo, 2014; Orozco-Galvan et al., 2015) to implement multi-standard OFDM modulators.

The first flexible hardware platform designed for multiple 5G waveform scenarios was proposed in (Nadal et al., 2018). It is a complete transmitter/receiver platform that comprises hardware and software modules for digital baseband processing, RF boards and high-level software applications for system control and information display purposes. Baseband processing for transmission and reception is implemented in two separate Xilinx Zynq devices. Three 5G waveform candidates are supported - OFDM, FBMC and UFMC. A high-level software application running on the ARM processor selects the type of waveform and the baseband parameters to be used. At the baseband processing level, flexibility comes from multiplexing between the three baseband modulators. For each waveform, two numerologies are supported by defining the value of specific baseband parameters.

The multi-mode baseband modulators just mentioned follow a *static multi-mode approach*. Through mux/demux structures and parameter value selection, the functional elements are selected and their mode of operation is modified according to the communication specifications. In a static multi-mode approach, hardware modules for all baseband processing operations are permanently present in the system, even if some of them are not always needed. Moreover, the functional elements are resource-inefficient because they are designed for the worst-case scenario. This may prevent the use of a smaller FPGA device, with a lower power consumption. Furthermore, the support for new modes of operation requires the redesign of the baseband processor from the scratch. As a consequence, these architectures are not upgradeable.

Another aspect has to do with non-contiguous CA scenarios. Multiple baseband processors

with possibly different parameterizations are necessary to independently process each component carriers (Yuan et al., 2010). Here, static multi-mode designs are inefficient, not scalable nor flexible. For instance, to perform spectrum aggregation of three component carries using OFDM, FBMC and UFMC, it would be necessary to replicate the multi-mode baseband modulator from (Nadal et al., 2018) three times. As each multi-mode modulator comprises one datapath for each waveform, nine datapaths would have to be implemented in the FPGA logic fabric. From those nine datapaths, only a maximum of three (a third) could be used at a time.

An alternative to the static multi-mode approach is the exploitation of run-time reconfigurability in FPGAs. The application of DPR in real-time FPGA-based systems was investigated by Pezzarossa et al. (Pezzarossa et al., 2017), through the analysis of resource utilization and the impact of DPR latency on system performance. Results indicate that, for computationally intensive tasks (e.g.: filter-banks or 32×32 matrix arithmetics), specialized circuits combined with DPR use FPGA resources more efficiently than a general (static) circuit, without compromising the computational performance. In wireless communications, Rousseau et al. (Rousseau et al., 2012) consider that flexible radio platforms can benefit from DPR advantages such as lower hardware constraints, hardware function optimization and higher flexibility. Kazaz et al (Kazaz et al., 2016) suggest the exploitation of DPR to adapt baseband processing chains of SDR systems.

2.3.1 Run-time Reconfiguration of FPGAs

This section addresses two techniques for run-time reconfiguration of FPGAs[1]: Dynamic Partial Reconfiguration (DPR) and Dynamic Frequency Scaling (DFS). The former allows for circuitry customization during operation, while the later uses the FPGA clock synthesis features to adjust the frequency of one or more clock signals on-the-fly.

In (Papadimitriou et al., 2011) a general architectural model for DPR (Figure 2.12) is presented to help the understanding of the operations intrinsic to DPR and the factors that influence DPR latency - the time required to reconfigure parts of the FPGA with one or more partial bitstreams. From a high level perspective, there are 3 main reconfiguration stages: 1) the bitstreams are copied from an *external memory* (the bitstream repository) to an FPGA *on-chip memory*; 2) the *reconfiguration controller* sends the bitstreams to the configuration port; 3) the bitstreams are loaded into the *configuration memory*, through the *configuration port*. Depending on the bandwidth of the memory elements involved, these stages are iteratively executed until all partial bitstreams are loaded into the configuration memory. Typically, stages 1 and 2 have a larger time overhead associated. Ultimately, the DPR latency is related to the size of the partial bitstream file(s) and the configuration port bandwidth.

In Xilinx FPGAs, the smallest reconfigurable region is defined as *reconfigurable frame* and it represents an area of 1-element (Configurable Logic Block (CLB)[2], Block RAM (BRAM) or DSP[3])

[1] Throughout this thesis, Xilinx terminology for 7-series devices is used when referring to FPGA technical aspects.

[2] In Xilinx 7-series FPGAs, every CLB contains two logic *slices* and is located next to a switch matrix. In turn, each slice is composed of 4 Lookup tables (LUTs) and 8 Flip-Flops (FFs).

[3] From now on, the acronym *DSP* refers to the dedicated DSP slices embedded on the FPGA logic fabric, rather than the generic *Digital Signal Processor* term.

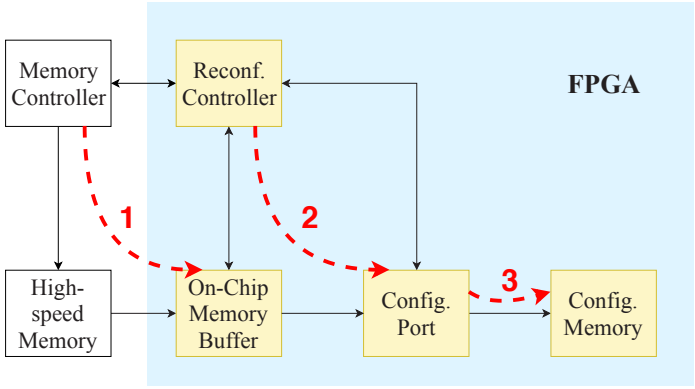


Figure 2.12: DPR general architectural model from (Papadimitriou et al., 2011). Numbers in red refer to the DPR stages: 1) copy bitstreams from external memory to on-chip memory; 2) send bitstreams from on-chip memory to configuration port; 3) bitstream writing on FPGA configuration memory.

wide by one clock region high (Xil, 2015b). For 7-series devices, reconfigurable frames are column-arranged and can represent 1×50 CLBs (width $\times$ height), 1×10 BRAMs or 1×10 DSPs. As DPR is performed frame-by-frame, the size of partial bitstreams depends on the amount and type of frames to reconfigure. A reconfigurable frame is viewed as an atomic unit and a partial bitstream can only address a discrete amount of frames. This does not mean that the boundaries of reconfigurable regions must conform with those of the reconfigurable frames. A reconfigurable partition (RP), i.e. a region defined as reconfigurable whose circuitry can be reconfigured at run-time, can have any element height, but its width and composition must not break FPGA interconnect columns. For instance, let us assume a Xilinx 7-series device and the three valid reconfigurable partitions depicted on Figure 2.13. RP1 and RP2 are 6-element wide with the same horizontal composition (4 CLBs, 1 BRAM and 1 DSP), but present different heights: RP1 is half a clock region high; RP2 is one clock region high. Consequently, RP2 has twice the area and reserved resources of RP1. Each column of RP1 reserves only half the resources addressed by the corresponding reconfigurable frame. However, the partial bitstream generated fro RP1 will contain information about the complete reconfigurable frames. In other words, the partial bitstream size for RP1 and RP2 would be equal, because the RPs have the same element width and horizontal composition and both vertically span over the same discrete amount of clock regions. Despite being only $1/5$ clock region high, the bitstream size for RP3 (horizontal composition: 6 CLBs, 1 BRAM, 1 DSP) would be larger than for RP1 and RP2. With the exception for dynamic elements (BRAMs, distributed LUT-RAMs or super logic regions), the non-reserved resources of the reconfigurable frames occupied by RP1 and RP3 can be used to place static logic. Thus, assuming that all three RPs have enough resources for a certain application, RP1 or RP3 would be chosen to reduce the overhead of DPR reserved resources, while RP2 would be chosen to increase the degrees of freedom and (potentially) the efficient of place-and-route procedures inside the RP. Despite irregular RP shapes (T or L shapes) are permitted, place-and-route inside those regions tend to be more challenging (Xil, 2015b).

Design-time decisions on system partitioning and reconfiguration granularity level determine

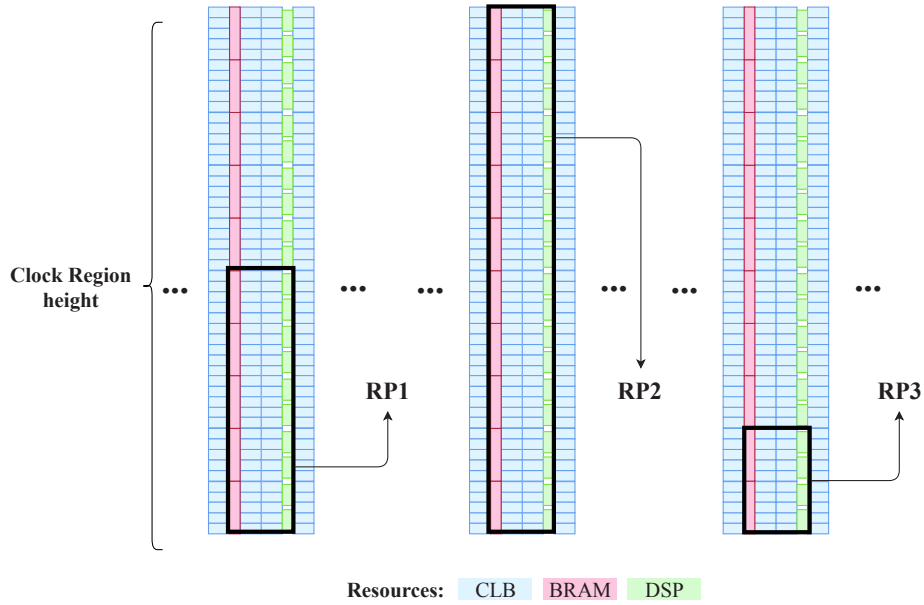


Figure 2.13: Reconfigurable partitions on Xilinx 7-series FPGA devices.

the amount/area of RPs and should consider the aspects just described. A finer granularity for reconfiguration would reduce the dynamically reconfigurable area, resulting in smaller bitstreams and DPR latency. However, if the granularity for reconfiguration is smaller than the smallest reconfigurable region within the FPGA, the design may use resource inefficiently. On the other hand, a coarser granularity for reconfiguration allows for the combination of several modules in the same RP, resulting in optimized floorplaning and better RP resource utilization. Yet, this would result in larger reconfigurable areas with larger partial bitstreams and associated DPR latency.

Bitstream compression can be used to reduce partial bitstream sizes and, consequently, DPR latency. Although several compression algorithms have been proposed in literature (Pan et al., 2004; Gu and Chen, 2008), they require bitstream decompression before writing on the FPGA configuration memory, which can negatively affect overall DPR latency. Xilinx EDA tools offer a compression option during bitstream generation. The compression algorithm writes identical configuration frames in multiple address locations at once, instead of writing them one by one. To do so, the bitstream uses the *Multiple Frame Write Register* (MFWR) to mark address locations with the same frame content (Xil, 2018). This algorithm does not require a decompression operation prior to reconfiguration and effectively helps in reducing DPR latency.

Another way to mitigate DPR latency is to enhance the configuration port throughput. In Xilinx devices, there are different configuration ports and those with higher bandwidths are the SelectMAP port and the *Internal Configuration Access Port* (ICAP). Both work in the same way, but the SelectMAP requires an external processor or programmable logic device to provide data and clock signal (Xil, 2016b). The ICAP is an internal interface that allows the FPGA to reconfigure itself, without external intervention. The ICAP data interface supports bit-widths of 8, 16 and 32, and Xilinx limits its maximum clock rate to 100 MHz (reconfiguration throughput of 400 MB/s) (Xil, 2015b). However, several published works have successfully overclocked the ICAP to achieve

higher reconfiguration speeds (Claus et al., 2010; Hansen et al., 2011). Another issue derives from the fact that the actual ICAP maximum clock frequency varies from device to device due to intrinsic manufacturing process variability (Vipin and Fahmy, 2018). According to (Hansen et al., 2011), the ICAP throughput scales linearly with the clock frequency and, for high ICAP bandwidths, the bottleneck becomes the provision of partial bitstreams from the external memory. This can be effectively accelerated through DMA techniques (Vipin and Fahmy, 2014). In Xilinx Zynq SoC devices it is also possible to access the FPGA configuration memory from the ARM processor, through the Processor Configuration Access Port (PCAP). The PCAP is a 32-bit interface that runs at 100 MHz and is controlled by a driver function. Thus, it does not need any type of FPGA logic fabric resource for DPR control purposes. Despite having theoretical maximum throughput equal to 400 MB/s, practical designs do not get closer to this limit due to the interconnection overhead in the ARM internal architecture. In this work, the ICAP is the preferred interface to access the FPGA configuration memory for two reasons: *1)* it provides higher bandwidths than the PCAP; *2)* its control can be customized by the designer and implemented on the FPGA logic fabric, without the need for an external interface.

Regarding energy overhead, Rousseau et al. (Rousseau et al., 2012) verified that the energy due to bitstreams fetching from external memory is much more significant than the energy associated with stages 2 and 3 from Figure 2.12. Consequently, reducing the partial bitstreams size not only reduces DPR latency, but also reduces the DPR energy overhead. The Hamming distance between the bitstreams from the previous and the next configuration are another impacting factor for the energy consumption behavior during DPR: more differences require a bigger reconfiguration effort Bonamy et al. (2014).

In the context of baseband processor design, DFS not only enhances dynamic power efficiency (Nunez-Yanez and Beldachi, 2014), but also augments the system flexibility by allowing the adjustment of processing throughput according to the communication demands. Clock management tiles (CMTs) in Xilinx 7-series devices include a mixed-mode clock manager (MMCM) and a phase-locked loop (PLL). These primitives operate in a similar way and can function as frequency synthesizers, jitter filters or deskew clocks (Xil, a). The MMCM has the particularity of supporting fine and dynamic phase shifting. A voltage controlled oscillator (VCO) is present in the MMCM/PLL and its frequency of operation is determined by the input reference clock and the values of two counters (M and D):

$$f_{VCO} = f_{clkIN} \times \frac{M}{D} \tag{2.12}$$

The VCO drives seven output counters (O_x), allowing the synthesis of seven different output clock signals:

$$f_{clkO_x} = f_{VCO} \times \frac{1}{O_x}, \qquad x = 0, 1, \ldots, 6 \tag{2.13}$$

Through the Dynamic Reconfiguration Port (DRP) of the MMCM or PLL primitives, it is possible to write configuration bits in order to change the values of M, D and O_x, as well as the phase of each synthesized clock. This can be done at run-time without loading a new bitstream into the

FPGA - *dynamic frequency scaling*.

2.3.2 Dynamically Reconfigurable Baseband Processors

The DPR application to baseband processing in wireless communications started with the adaptation of small-scale and relatively simple functional elements such as FIR filters, constellation mappers or channel encoders (Delahaye et al., 2007; Delorme et al., 2008). One of the first multi-waveform flexible PHY architectures was proposed by He et al. (He et al., 2011). It is an SDR architecture implemented on a Xilinx Virtex-5 FPGA device and combines two reconfiguration techniques: *a)* DPR is employed to dynamically change the baseband processing mode of operation (e.g.: FFT size, modulation scheme and CP length); and *b)* DFS is used to adapt the clock frequency of the digital up-converter and the baseband processor. The design supports two waveforms (OFDM and WCDMA) and several 3G/4G standards and modes of operation. Compared with a static multi-mode design, the DPR-based design achieved a reduction in the number of slices, DSP blocks (DSPs) and block RAMs (BRAMs) used. However, the comparison is not fair as the static multi-mode design considers parallel and independent processing chains for each standard, ignoring potential optimization coming from the reutilization of common functional elements across different standards. Moreover, this work lacks a comprehensive analysis of the run-time reconfiguration energy/latency overheads and an evaluation of the DFS impact on power consumption.

Vipin and Fahmy (Vipin and Fahmy, 2015) propose CoPR, an automated framework for DPR-based adaptive systems on a Xilinx Zynq device. An illustrative case study is presented, where a reconfigurable multi-standard baseband OFDM transmitter is designed. The design supports three standards (IEEE 802.11, IEEE 802.16 and IEE 802.22) and considers two RPs: one to implement the digital modulation scheme (QPSK, 16-QAM or 64-QAM) and another to accommodate the OFDM processing datapath. DPR is performed through the Zynq PCAP interface, which leads to lower reconfiguration speeds. This work only reports reconfiguration time results and does not provide figures for power consumption or RP reserved resources.

Rihani et al. (Rihani et al., 2017) present an ARM-FPGA-based platform implemented on a Xilinx Zynq device. Several processes run on the ARM processor and retrieve communication environment information. In turn, this information is used to instruct a configuration controller about how to reconfigure an OFDM baseband processing modulator. An OFDM transmitter supporting WiFi and WiMAX is implemented on Zynq's programmable logic and features four RPs used for scrambling, interleaving, FEC encoding and IFFT. As in (Vipin and Fahmy, 2015), DPR is achieved via the PCAP interface. Results for the transmitter resource utilization and DPR latency are presented. The power consumption for the Zynq's processing system during DPR is measured with the Power Measurement Bus (PMBus) and the TI Digital Power Design software (TIf). However, the sampling period of the measurements is within the order of magnitude of the reconfiguration times observed (ms). Thus, the measurement method is not accurate or suitable for the real-time measurement in the time intervals of interest.

Also exploiting a hybrid Zynq FPGA platform, (Shreejith et al., 2015) present a HW/SW co-design for CR systems combinig parameter reconfiguration and DPR. To illustrate their approach, a DVB baseband processor supporting DVB-cable (DVB-C) and DVB-satellite (DVB-S) is implemented. The software system domain (ARM processor core) is responsible for running a cognitive algorithm and for controlling the reconfiguration of the baseband processor implemented in the hardware domain (FPGA logic fabric). Only DPR latency and RP reserved resources are reported. The access to the FPGA configuration memory is done through the ICAP and a DMA controller used to accelerate the bitstreams transfer from the external memory to the ICAP (Vipin and Fahmy, 2014).

Pham et al. (Pham et al., 2017) present a reconfigurable multi-standard OFDM transceiver supporting IEEE 802.11, IEEE 802.16 and IEEE 802.22 on a Xilinx Virtex-6 FPGA. The modulator side is implemented in a single RP, whereas the demodulator explores a mixture of DPR-based and static multi-mode modules. The access to the FPGA configuration memory follows the approach from (Vipin and Fahmy, 2014). The authors only present reconfiguration time and bitstream size figures. More emphasis is put on the receiver processing chain architecture, impact of DPR in the halting time and FIFO buffering requirements in this application domain.

Table 2.3 summarizes the results for DPR impact in the reconfigurable baseband processors from (He et al., 2011; Vipin and Fahmy, 2015; Shreejith et al., 2015; Pham et al., 2017; Rihani et al., 2017). In terms of application, these works are devoted to 3G/4G standards and waveforms. From a system level perspective, they focus primarily on the enhanced flexibility DPR can offer, paying less attention to the overall impact of this technique on the design of hardware infrastructures for wireless communications. Most of these works present only results for resource utilization and DPR latency, often ignoring power consumption. There is a lack of fair comparison between functionally equivalent static multi-mode and DPR-based designs, in order to conveniently discuss design trade-offs. In this dissertation, section 5.1.3 provides an extensive comparison between a static multi-mode and DPR-based design, as well as a quantitative analysis of DPR latency and energy oerhead. None of the mentioned works supports presents a architecture with multiple and independent processors suitable for non-contiguous spectrum aggregation (section 5.3 of this dissertation).

Table 2.3: Summary of related works on reconfigurable baseband modulators exploring DPR techniques and respective results for DPR impact on the system

	Waveforms	FPGA device	DPR interface	# RPs	RP reserved resources	DPR latency (worst-case)	DPR speed	Bitstream size per RP
(He et al., 2011)	OFDM WCDMA	xc5vlx110t	n/a	2	n/a	n/a	n/a	292 kB; 282 kB
(Vipin and Fahmy, 2015)	OFDM	xc7z020	PCAP	2	n/a	5.18 ms	130 MB/s (c)	14.2 kB; 675.4 kB
(Rihani et al., 2017)	OFDM	xc7z020	PCAP	4	n/a	2.118 ms	128 MB/s (c)	30 kB; 30 kB 180 kB; 30 kB
(Shreejith et al., 2015)	SC-QAM OFDM	xc7z020	ICAP + DMA	1	LUTs: 5400 FFs: 8000 BRAMs: 50 DSP: 40	768 µs	380 MB/s (tp)	305 kB (c)
(Pham et al., 2017)	OFDM	xc6vlx240t	ICAP + DMA	Tx: 1 Rx: 2	n/a	Tx: 642 µs (c) Rx: 1.5 ms (c)	380 MB/s (tp)	Tx: 250 kB Rx: 200 kB + 370 kB

n/a: not available/applicable; *(c)*: calculated, not measured; *(tp)*: theoretical peak value

Chapter 3

FFT Cores Implementation

This chapter addresses the FPGA-based implementation of FFT cores. As highlighted in Section 2.1, the synthesis and analysis for the considered waveforms is efficiently implemented using IFFT/FFT cores. The considerable computational complexity of the FFT motivates careful considerations during hardware implementation.

OFDM and FBMC perform well for large packet bursts (Parvez et al., 2018), where a continuous stream of multicarrier symbols has to be processed. In particular, FBMC has a high time domain overhead (Kibria et al., 2016), which prevents its application for short packet transmission. In turn, UFMC performs well for short packet lengths and sporadic burst transmission (Schaich et al., 2014; Parvez et al., 2018). Another aspect to consider in UFMC is that the parallel subband processing requires an IFFT core per branch. From these observations, high-performance pipelined FFT architectures were adopted in OFDM and FBMC datapaths, while low-resource memory-based FFT architectures were adopted in the UFMC datapath.

The pipelined and memory-based FFT core designs presented in this chapter are implemented on a Xilinx xc7z020 device. Both input and output ports of the FFT cores follow the AXI4-Stream interface protocol and all arithmetic computations are done in fixed-point precision: 32-bit wide datapath, with 16 bit for both real and imaginary parts (Q5.11 format).

3.1 Pipelined FFT cores for OFDM and FBMC

Prior to a decision on the FFT algorithm to adopt, typical FFT sizes used in OFDM and FBMC systems were analyzed. Regarding OFDM, the LTE standard for downlink transmission was considered. It defines 6 modes of operation whose FFT sizes are the powers of two from 128 to 2048, as well as 1536 (Ghosh and Ratasuk, 2011). Currently, there are no FBMC-based standards. In (Nadal et al., 2016; Berg et al., 2014), the FFT sizes considered for FBMC baseband processors are 2048 and 4096. Table 3.1 lists the considered FFT sizes and respective factorizations for OFDM and FBMC datapaths. The factorizations are in the form $4^x.2^y.3^z$, with $x \in \{3,4,5,6\}$, $y \in \{0,1\}$ and $z \in \{0,1\}$. The exponents x, y and z indicate the number of Radix-4, Radix-2 and Radix-3 stages, respectively. In this implementation, Radix-4 stages are implemented using the Radix-2^2

Table 3.1: Implemented FFT sizes and their factorization

FFT Size	Factorization	# Radix-4 stages	# Radix-2 stages	# Radix-3 stages
128	$4^3 \times 2$	3	1	-
256	4^4	4	-	-
512	$4^4 \times 2$	4	1	-
1024	4^5	5	-	-
1536	$4^4 \times 2 \times 3$	4	1	1
2048	$4^5 \times 2$	5	1	-
4096	4^6	6	-	-

algorithm (He and Torkelson, 1996), in order to reduce the butterfly complexity (Section 2.2). Thus, the algorithm adopted here is the *Cooley-Tukey Mixed-Radix*-$2^2/2/3$ algorithm. The choice for a pipelined architecture mainly relies on the overall trade-off between resource utilization, throughput and control complexity. In this work, the Single-Path Delay Feedback (SDF) is the preferred one, because of its simpler implementation, lower memory requirements and acceptable throughput.

To better understand the composition of the pipelined architecture for each FFT size, let us first consider the 256-FFT case (Figure 3.1). From a high-level perspective, it comprises four ($\log_4 256$) Radix-2^2 processing elements (PEs), whose operation is dictated by the *Radix*-$2^2/2$ *Control Unit*. This unit feeds the input data to the subsequent pipeline elements and generates a $\log_2 N$-bit binary counter that controls the operation of each Radix-2^2 PE. Additionally, the binary counter is used to fetch the correct twiddle factors for the complex multiplications between Radix-2^2 PEs. As *decimation in frequency* is adopted, a bit-reversal reordering operation is required to provide the output results in natural order.

A Radix-2^2 PE is composed of two Radix-2 butterflies and each of them has a feedback shift register. The number associated with the shift register is the amount of elements stored in it. These two butterflies are not exactly the same, hence the distinction between *R2 BF-I* and *R2 BF-II*. Figure 3.2 illustrates their internal structure. The main difference between them is the multiplexing between the *x[1]* and $x[1].(-j)$. In fact, this multiplication consists of swapping the real and imaginary parts of *x[1]* and invert the imaginary part sign. Apart from this, the operation of both butterflies is very similar. They have two operation stages: when *sel* is *'0'*, the butterfly is kept idle and the input data is forward to the feedback register, while the its contents are flushed to the output; when *sel* is *'1'*, the butterfly is active, the addition result is forward to the output and the difference result is fed back to the shift register. The *sel* signal is a bit from the control counter generated by the *Radix*-$2^2/2$ *Control Unit*. For a given Radix-2^2 PE, the *swap* bit for the *R2 BF-II* butterfly is the *sel* bit from the corresponding *R2 BF-I* butterfly. The addressing scheme for the twiddle factor ROMs between Radix-2^2 PEs follow the work from (Garrido et al., 2013).

According to the Table 3.1, a 512-FFT (Figure 3.3) can be obtained by adding a Radix-2 PE to the pipeline structure that implements a 256-FFT. As this involves two different radixes - 2^2 and 2-, the *Radix*-$2^2/2$ *Control Unit* has to apply the two-dimension index mapping from eqs. 2.7, 2.8 on

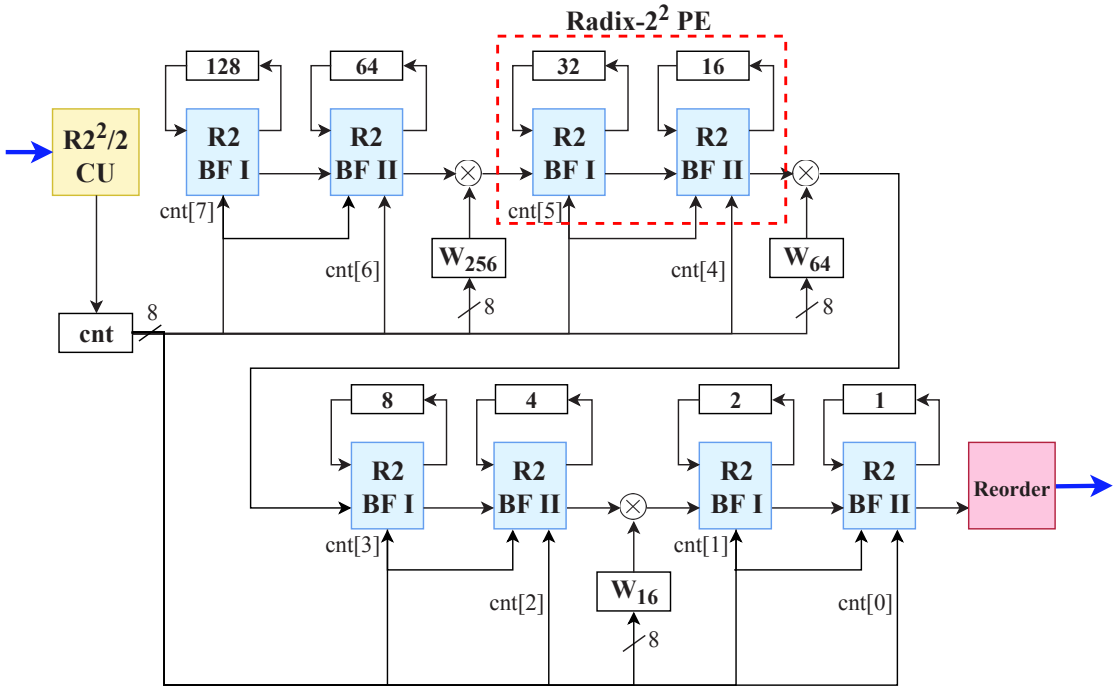
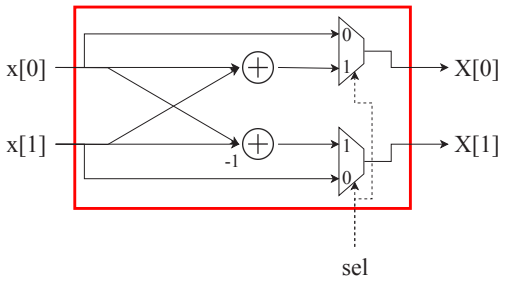
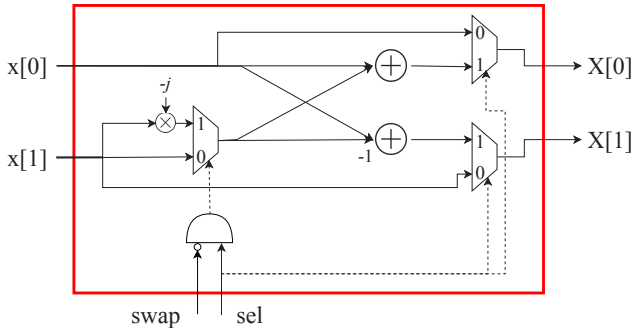


Figure 3.1: Pipelined architecture for the 256-FFT core.



(a) Type I



(b) Type II

Figure 3.2: Radix-2 butterfly architectures.

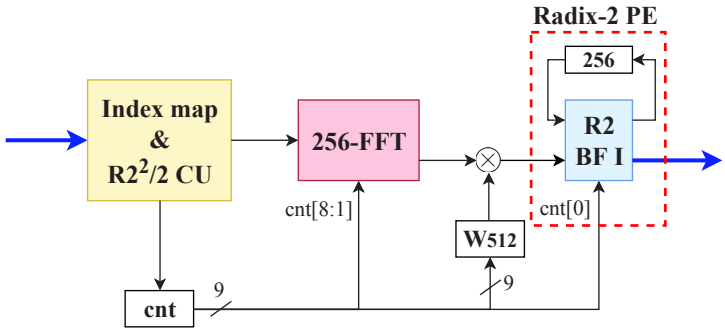


Figure 3.3: Pipelined architecture for the 512-FFT core.

the input data. Additionally, *inter-radix rotations* are required between the the radix domains - the term $W_N^{n_2 k_1}$ in eq. 2.9. Due to the index mapping at the input, the bit-reverse reordering can be kept in the Radix-2^2 domain and no further reordering is required to obtain the output in natural order. The Radix-2 PE internal structure correspond the *R2 BF-I* in the Radix-2^2 PE and it is controlled by the least significant bit of the binary counter generated from the Radix-$2^2/2$ Control Unit.

Similarly, a 1536-FFT (Figure 3.4) is built by adding a Radix-3 PE to the 512-FFT processing chain. Again, index mapping is required at the input and an additional *inter-radix rotation* is needed between the Radix-2 and Radix-3 PEs. So far, the SDF architecture has been applied to the Radix-2 and Radix-2^2 PEs. Its feedback nature requires the butterfly operations to be executed in a single stage. In other words, the datapath inside the butterfly cannot be pipelined with intermediate registers, under penalty of corrupting the normal SDF operation. This limitation is irrelevant for simple butterfly structures, whose operations can be easily executed in a single stage. The Radix-3 butterfly is more complex. From eq. 2.5, we have:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \end{bmatrix} = \begin{bmatrix} e^0 & e^0 & e^0 \\ e^0 & e^{-j\frac{2\pi}{3}} & e^{-j\frac{4\pi}{3}} \\ e^0 & e^{-j\frac{4\pi}{3}} & e^{-j\frac{8\pi}{3}} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \end{bmatrix} \tag{3.1}$$

For low clock frequencies, 3.1 can be executed in a single stage. However, if higher throughput is required, it may be necessary to partition the datapath of the Radix-3 butterfly in several stages. As a result, a Multi-path Delay Commutator (MDC) architecture was adopted in Radix-3 PE. Its

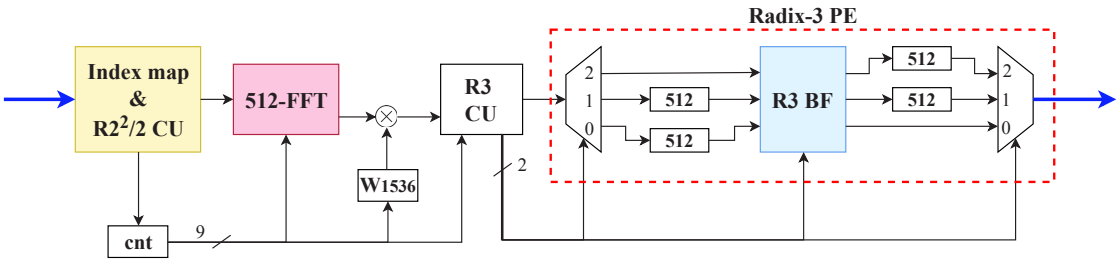


Figure 3.4: Pipelined architecture for the 1536-FFT core.

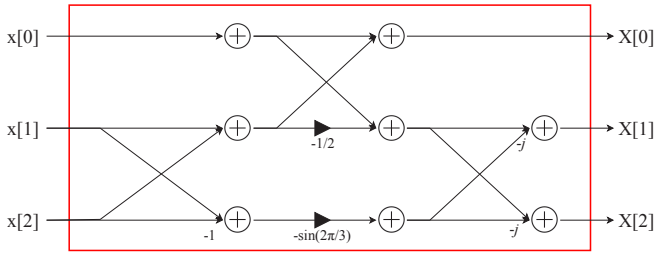


Figure 3.5: Signal flow graph for the Radix-3 butterfly.

architecture is visible in Figure 3.4 and the signal flow graph for the Radix-3 butterfly is depicted in Figure 3.5. To achieve higher clock frequencies, the Radix-3 butterfly datapath was divided in three stages, as presented in (Löfgren and Nilsson, 2011):

Stage 1:	*Stage 2:*	*Stage 3:*
$a_0 = x[0]$	$b_0 = a_0 + a_1$	$X[0] = b_0$
$a_1 = x[1] + x[2]$	$b_1 = a_0 + \frac{a_1}{2}$	$X[1] = b_1 + j.b_2$
$a_2 = x[1] - x[2]$	$b_2 = -\sin(\frac{2\pi}{3}).a_2$	$X[2] = b_1 - j.b_2$

In contrast with the Radix-2^2 and Radix-2 PEs, the Radix-3 PE operation cannot be controlled by a simple binary counter. The *Radix-3 Control Unit* is an FSM with three states that works similarly to (Cho et al., 2013). During the S_{init1} state, the butterfly is idle and the demux forwards the incoming data to its output *0*, filling the corresponding shift register. At the same time, the contents of the shift register connected to the input *0* of the multiplexer are flushed. Then, the S_{init2} state performs similar operations on the shift registers connected to the input/output 1 of the demux/mux structures. During the S_{proc} state, the Radix-3 butterfly picks and process the input data and the data stored in the shift registers. The results are sent to the feed-forward shift registers and the multiplexer. Each FSM state lasts for 512 active clock cycles.

For the remaining FFT sizes, the FFT pipeline structure can be inferred from these examples and the factorizations listed in Table 3.1. The pipelined architectures presented in this section are labeled as *FFT cores*, but they are also used as IFFT cores, following the scheme from Figure 2.11. The only relevant modification on the IFFT cores is the output scaling by $1/2$ in every *R2 BF I*/*R2 BF I* and $1/3$ in the Radix-3 butterfly. While the scaling by $1/2$ is actually a 1-bit shift-right operation, the scaling by $1/3$ may involve additional non-trivial multiplications in the 1536-IFFT. An FPGA-based FFT core for each considered size was implemented.

After the pipelined initialization latency, the FFT cores enter in steady-state operation and are able to produce a throughput of one sample per clock, assuming the input data vectors arrive continuously. Thus, the processing throughput is mainly dictated by the clock frequency. The initialization latency varies with the FFT size: it is 348 clock cycles for 128-FFT and 12326 clock cycles for 4096-FFT. Post Place-and-Route results for resource utilization are presented in Table 3.2, for a 100 MHz clock frequency. In general terms, resource utilization increases with the FFT size. The 1536-FFT core consumes more flip-flops (FFs), whereas the 4096-FFT core shows a higher slice, LUT and BRAM utilization. The DSP utilization is higher for 1536-FFT, 2048-FFT and

Table 3.2: Resource utilization for the implemented FFT cores; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz

Resource	Available	FFT size						
		128	256	512	1024	1536	2048	4096
Slice	13300	614	598	791	855	1149	994	1432
LUT	53200	1675	1888	2245	2678	3320	3024	4307
FF	106400	1073	964	1246	1162	1962	1444	1384
BRAM	140	2	3	5.5	7.5	11	12	25.5
DSP	220	9	9	12	12	15	15	15

4096-FFT due to the existence of more Radix-2^2 stages and, in the 1536-FFT case, more inter-radix rotations and non-trivial multiplications inside the Radix-3 butterfly. In turn, 4096-FFT deals with a larger number of data samples, leading to higher BRAM demands for data storage and reordering along the processing pipeline. The resource usage variation between the smaller and larger FFT sizes is considerable and suggests that a multiplexer-based approach to simultaneously implement all FFT sizes would result in resource waste. Hence, employing circuit run-time specialization through DPR, could cover the near-instantaneous system demands and potentially lead to a better resource efficiency.

In pipelined FFT architectures, the operation of all pipeline stages is tightly synchronized and pruning an operation at one stage corrupts the operation of the other stages. Although it is impossible to apply FFT pruning to pipeline FFT architectures, we investigated the impact of data sparsity in CR and DSA scenarios, where NC-OFDM or FBMC can be applied. Towards this end, a *sparse data* variant for the pipelined FFT core is presented. The difference between the sparse data variant and baseline FFT core is the design of the complex multipliers used for twiddle factor rotations. The algorithm adopted for complex multiplication follows the approach exposed in (Meyer-Baese, 2007), which decomposes a complex multiplication in three real multipliers, one adder, and two subtractors. For the sparse data variant, PE results before the complex multipliers are inspected and, if zero, a *zero flag* is activated on the input of both complex multiplier and twiddle factor ROM. This flag will prevent the ROM output and the registers within the complex multiplier from switching their values during a multiplication by zero. Figure 3.6 illustrates the architecture of the complex multiplier adopted and highlights the differences between its baseline and sparse data variant.

The goal is to investigate if, in sparse data scenarios, the modification on the complex multiplier produces a significant node activity and power consumption reduction on the FFT datapath. There is no impact on throughput or initialization latency. The sparse data variant requires extra logic to detect zero values at the complex multiplier input and to propagate the zero flag. However, this extra logic represents a small amount of resources: at most 76 LUTs and 31 FFs. Two scenarios were considered for power estimations: *dense data* and *sparse data*. The former considers an input array of non-zero data subcarriers which are processed by the FFT core. In the later, the input data profile only has 6.25% of non-zero data subcarriers. The elaboration of this data profile was

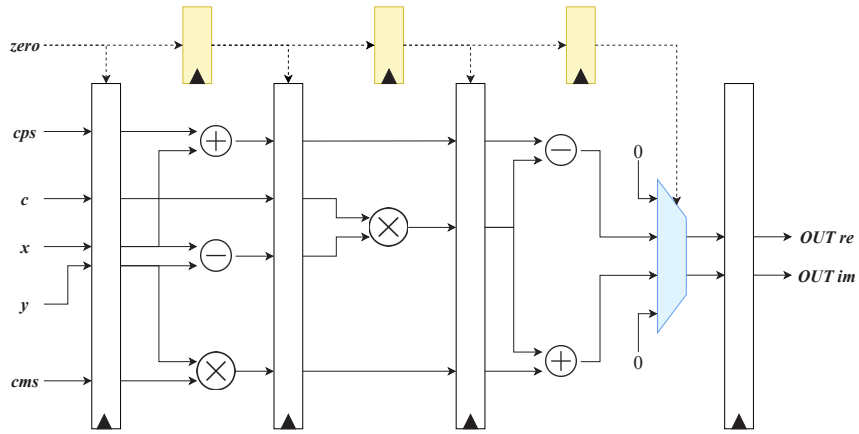


Figure 3.6: Complex multiplier architecture. The circuit produces the multiplication between two complex factors: $A = x + jy$ and $B = c + js$. The factor B is a pre-computed constant stored as a triplet $\{c; cps = c + s, cms = c - s\}$ (Meyer-Baese, 2007). The dashed-lined elements are only present in the OFDM sparse data variant.

inspired on the FFT input data profiles presented in (Airoldi et al., 2010). Tables 3.3a and 3.3b present power estimates for the baseline FFT cores and their sparse data variants, respectively.

Comparing Tables 3.3a and 3.3b, static power shows no variation, regardless FFT variant or input data scenario. This is to be expected on an FPGA, because the circuit active area and estimated junction temperature are almost the same. In turn, dynamic power consumption increases with the FFT size. In particular, 4096-FFT presents a dynamic power consumption that is more than three times the one observed for 128-FFT. This observation further motivates the application of run-time circuit specialization on the FFT cores. The data sparsity also impacts the dynamic power and, for a given FFT core implementation, it consumes between 20 mW to 30 mW more in the dense data scenario. However, the results do not evidence a clear advantage for adopting the sparse data variant of the FFT cores. Both FFT variants exhibit almost the same dynamic power consumption, regardless the input data profile. The maximum dynamic power variation between the baseline FFT cores and their sparse data variant is only 5 mW, favoring the baseline 1536-FFT in the dense data scenario and the sparse 4096-FFT variant in the sparse data scenario. This reveals that the modifications introduced by the sparse data variant do not bring power benefits. Consequently, in this work, the baseline FFT cores will be used in the implementation of OFDM and FBMC baseband datapaths.

3.2 Memory-Based FFT core for UFMC

As stated earlier in this chapter, the UFMC modulator features an memory-based IFFT core per active subband. Similar to FBMC, no UFMC-based communication standards have been released so far. The works from (Nadal et al., 2018; Jafri et al., 2018; Medjkouh et al., 2017) consider powers of two values for the number of available subcarriers (N), in particular 256, 512 or 1024. In the classic UFMC modulator approach, N would correspond to the IFFT size. Instead, this work

Table 3.3: Power consumption estimates for the FFT cores. Device: xc7z020; $f_{clk} = 100MHz$; Analysis tool: Vivado 2015.2; Post Place-and-Route power analysis with high confidence level; Node activity derived from post Place-and-Route simulation.

(a) Baseline FFT cores

	Static power (mW)		Dynamic Power (mW)	
	dense data	sparse data	dense data	sparse data
128-FFT	256	256	72	51
256-FFT	257	255	80	57
512-FFT	258	257	102	77
1024-FFT	259	258	120	95
1536-FFT	261	260	151	125
2048-FFT	261	260	150	123
4096-FFT	266	265	223	196

(b) Sparse data variant

	Static power (mW)		Dynamic Power (mW)	
	dense data	sparse data	dense data	sparse data
128-FFT	256	255	73	49
256-FFT	257	255	82	55
512-FFT	258	256	102	75
1024-FFT	259	258	123	96
1536-FFT	261	260	156	127
2048-FFT	261	259	149	121
4096-FFT	266	264	221	191

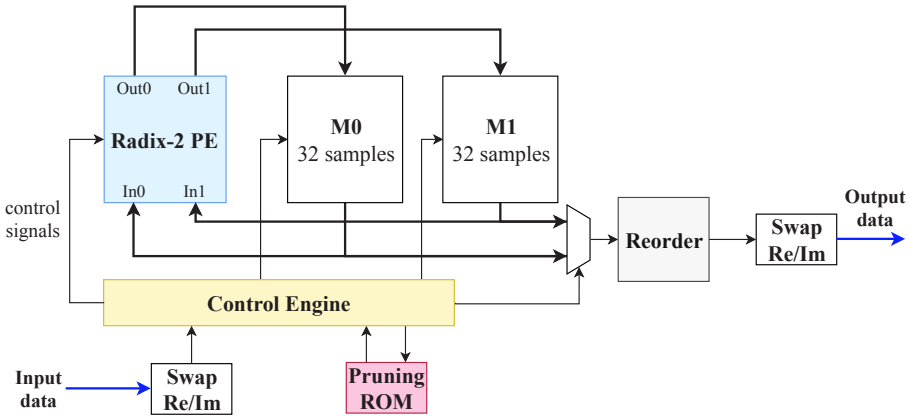


Figure 3.7: Memory-based IFFT architecture used in the UFMC baseband modulator

follows the approach from (Knopp et al., 2016), where a reduced N'-IFFT is combined with $\frac{N}{N'}$ upsampling. So, it is possible to support several values of N by keeping the same N'-IFFT core and adjusting the upsampling factor. The dimensioning of N' and the upsampling factor affects the OOB performance and should be carefully chosen. From the guidelines in (Knopp et al., 2016), the N' value considered here is 64. To reduce both resource utilization and control complexity of the 64-IFFT core, the memory-based architecture only has a single PE and employs the *Cooley-Tukey Radix-2* algorithm.

In a Radix-2 memory-based 64-IFFT core, the processing is divided into $\log_2 64 = 6$ processing stages of $64/2 = 32$ processing steps. However, as discussed in Section 3, memory-based FFT architectures can support pruning algorithms, thus reducing the overall amount of processing steps required. Typically, the amount of subcarriers per PRB (PRB size) in UFMC systems follows the LTE numerology, where a PRB spans over 12 subcarriers. For the considered case, their location within the IFFT input data array is known in advance: the 12 PRB subcarriers are mapped to the central bins of a 64-element array and the remaining 52 elements are zero. Following the DIF Radix-2 algorithm it is possible to pre-determine the *processing steps* that need to be executed and those that can be pruned. In this case, the first and second processing stages can be pruned and comprise only 12 and 24 processing steps, respectively. The remaining 4 processing stages cannot be pruned and require the execution of 32 steps each. Thus, FFT pruning allows the reduction of the total of processing steps from 192 (6×32) to 164 ($12 + 24 + 4 \times 32$).

The implemented memory-based IFFT architecture follows the control structure and address generation scheme from (Xiao et al., 2008a), but further extends it to support FFT pruning. The architecture is depicted in Figure 3.7 and its main constituent elements are: a control engine, a Radix-2 PE, two 32-element memory banks (*M0* and *M1*) and a ROM memory used to control IFFT pruning - *pruning ROM*. As decimation in frequency (DIF) is employed, a reordering unit is attached to deliver IFFT output results in natural order.

The control engine is the brain of the IFFT core, generating all the signals to control the Radix-2 PE and address the memories within the architecture. These control signals come from a 8-bit counter, whose value can be viewed as an opcode for each processing step. The counter

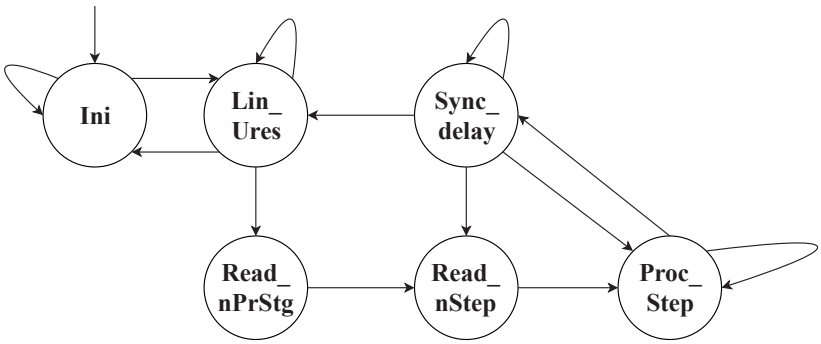


Figure 3.8: State diagram for the control engine FSM.

increment and step are controlled by an FSM with the state diagram represented in Figure 3.8. After initialization (*Ini* state) the FSM moves to the *load input/unload results* state (*Lin_Ures*). At this state, the control engine issues read/write operations on *M0* and *M1* to fill them with the incoming data samples, while forwarding the results from previous transform processing to the reordering unit.

Having received the complete input data vector, the control engine inspects the pruning ROM to see if there are any processing stages where pruning occurs - *pruning stages*. The first pruning ROM word is the number of pruning stages. The rest of the pruning ROM consists of blocks with the information about each pruning stage. The first word of these blocks is the number of processing steps in a given pruning stage and the remaining block words are the sequence of counter values to control those processing steps. In our case, there are 2 pruning stages comprising 12 and 24 processing steps each. So the pruning ROM is made of 39 words (8-bit each): one to indicate the amount of pruning stages, two to indicate the amount of processing steps of each pruning stage, and the $12+24=36$ counter values for each processing step.

After the completion of the *Lin_Ures* state, the control engine reads the number of pruning stages from the ROM (*Read_nPrStg*). For each pruning stage, the control engine reads the number of processing steps (*Read_nStep*) and proceeds with the *process transform* state (*Proc*). This state corresponds to the execution of the processing steps of each Radix-2 IFFT stage. The control unit fetches values from *M0* and *M1* and send them to the Radix-2 PE that performs a processing step. Then, the results are stored back in *M0* and *M1*. Each *Proc* state iteration is controlled by the counter value: for the pruning stages, the counter values are fetched from the pruning ROM; whereas for non-pruning stages, the control engine internally increments the counter from 0 to 31 $(64/2-1)$. When all *Proc* iterations of a processing stage are executed, the FSM moves to a synchronization state (*Sync_delay*). It introduces a waiting delay to synchronize the IFFT core operation. This is required due to the fact that the Radix-2 PE execution is internally pipelined and takes six clock cycles to complete.

The Radix-2 PE architecture is shown in Figure 3.9. The butterfly structure is similar to *R2 BF I* (Figure 3.2a). As IFFT is considered here, the only difference is the shift-right operation (scaling by 1/2) on the butterfly outputs. The mux-structures either forward or swap the PE inputs/outputs,

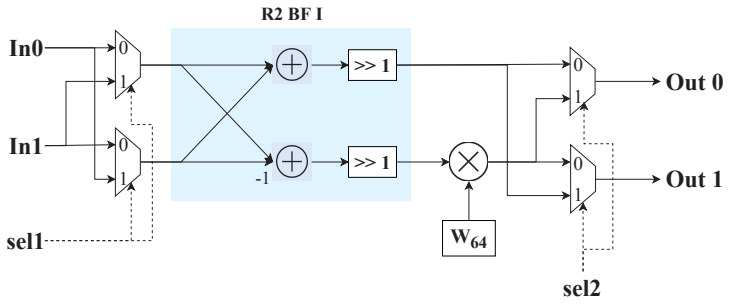


Figure 3.9: Radix-2 PE internal structure for the memory-based IFFT architecture

according to (Xiao et al., 2008a), and their control signals (*sel1* and *sel2*) are generated by the control engine. The complex multiplier architecture is similar to the one used for the baseline pipelined FFT cores. It performs the multiplication between the bottom butterfly output and a twiddle factor pre-stored in a ROM. Exploiting the Radix-2 algorithm and trigonometric symmetries, the amount of stored twiddle factors is reduced from 32 to 16.

In contrast with the pipelined FFT architectures, the burst-mode operation of the memory-based 64-IFFT core does not allow the processing of continuous data streams. This drops the processing throughput performance of the IFFT core. For the implemented 64-IFFT core, the latency is 342 clock cycles and the measured throughput is limited to around 0.2 samples per clock cycle. As expected, the resource utilization for the 64-IFFT core (Table 3.4) and its dynamic power consumption (19 mW) are considerably low. This is advantageous in the context of UFMC baseband modulation, as the IFFT core replication for each subband does not incur a heavy area and power penalty.

3.3 Summary

Due to the relevance of FFT/IFFT cores in multi-carrier modulation techniques, this chapter can be viewed as a starting point to the implementation of baseband processors. The architectural choices made during FFT/IFFT implementation were motivated by the main characteristics of the addressed waveforms and the presented results corroborate the trade-offs between FFT architectures highlighted in Section 2.2.

For OFDM and FBMC baseband processing, pipelined FFT cores following a Mixed-Radix-$2/2^2/3$ were implemented. The pipelines show a regular structure that scales with the FFT size.

Table 3.4: Resource utilization for the memory-based 64-IFFT core; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz

Resource	Slice	LUT	FF	BRAM	DSP
Available	13300	53200	106400	140	220
64-IFFT	170	533	503	2	3

Their capability for continuous data processing results in higher throughput and is well-suited for long sequence transmission scenarios. Resource utilization and power consumption generally increase with the FFT size. This suggests that the FFT/IFFT cores in OFDM and FBMC systems are a good candidate for the application of run-time circuit specialization. The modifications on the complex multipliers aiming at node activity reduction did not show consistent dynamic power benefits, even in sparse data scenarios.

In turn, the IFFT core for UFMC modulation follows a Radix-2 memory-based architecture. As UFMC is more suitable for short-burst and sporadic transmissions, the requirements for continuous processing and high throughput can be relaxed. Instead, the implemented memory-based IFFT core prioritizes the requirements on low resource utilization and dynamic power consumption. This reduces the burden of IFFT core replication for each UFMC subband.

Chapter 4

Implementation of Datapaths for Baseband Processing

Starting from the design of FFT cores presented in the previous chapter, this dissertation proceeds with the FPGA-based implementation of baseband processing datapaths for OFDM modulation and demodulation, FBMC and UFMC modulation. After presenting the architecture and operation of each datapath, their resource utilization for different numerologies is analyzed, in order to identify cases that would potentially benefit from run-time circuit specialization in the context of multi-mode baseband processors. Run-time circuit specialization is well-suited for baseband modules whose resource utilization shows considerable variations for different baseband parameters or whose internal structure can't be adapted by setting baseband parameter values. Special emphasis is put on the implemented FS-FBMC baseband modulator and its comparison with PPN-FBMC designs, as it is one of this dissertation's main contributions. All baseband processors presented in this chapter have been implemented on a Xilinx xc7z020 device. To facilitate the continuous data stream processing the input/output ports of all datapaths and their constituent modules follows the AXI4-Stream protocol. The arithmetic operations involved in baseband processing are performed with real and imaginary parts represented in Q5.11 format (16-bit).

4.1 OFDM Baseband Processing

OFDM-based communication standards define different numerologies that influence baseband processing and, consequently, their hardware implementation. In this dissertation, the numerologies considered for OFDM baseband processing are those defined for LTE downlink transmission. Time domain LTE signals are structured into *frames* of 10 ms. A frame comprises ten *subframes* of 1 ms and each subframe is composed of two *slots* with a duration of 0.5 ms. Slots consist of seven (for normal CP length) or six (for extended CP length) OFDM symbols. The smallest physical resource in LTE - *resource element* (RE) - consists of one subcarrier during one OFDM symbol. The REs are grouped into *resource blocks* (RBs), which represent the smallest resource units that can be allocated to a user. An RB comprises twelve consecutive subcarriers in the frequency domain and

Table 4.1: LTE baseband requirements for downlink transmission (Ghosh and Ratasuk, 2011). N_{RB}: number of resource blocks; A: number of active subcarriers (data+pilots) per OFDM symbol; N: IFFT/FFT size; L_{CP}: cyclic prefix length; W: number of time-domain samples over which WOLA is applied

Mode of operation	Sampling rate (MSample/s)	N_{RB}	A ($N_{RB} \times 12$)	N	L_{CP} per slot symbol		W
					1^{st} symbol	others	
$B_{1.4}$	1.92	6	72	128	10	9	4
B_3	3.84	15	180	256	20	18	6
B_5	7.68	25	300	512	40	36	4
B_{10}	15.36	50	600	1024	80	72	6
B_{15}	23.04	75	900	1536	120	108	8
B_{20}	30.72	100	1200	2048	160	144	8

one slot in the time domain. According to the LTE mode of operation, some OFDM parameters may vary and influence the baseband processing operations described in Section 2.1.1. Table 4.1 presents the baseband parameters for the six LTE downlink transmission modes of operation, considering OFDM symbols with normal CP length. For sake of simplicity, each mode of operation is designated by B_X, where X is the corresponding channel bandwidth in MHz. Regarding the digital symbol (de)modulation, the typical constellation schemes used in LTE are QPSK (4-QAM), 16-QAM and 64-QAM.

Although only one standard is considered here, it serves the purpose of introducing variability of baseband parameters and thus, allows for the study of flexible and reconfigurable baseband processors for multi-mode communications. Moreover, the adaptation of the presented datapaths for other OFDM-based standards is straightforward.

As previously mentioned, both OFDM modulation and demodulation are studied. Regarding the communication channel between them, a simple model for a good quality channel was considered. This model allows for the validation of the algorithms employed in the OFDM demodulation. The channel introduces a time delay between 0 and $N-1$ samples, a carrier frequency offset between 0 and 7.5 kHz (half Δf) and AWGN noise ($SNR = 22\ dB$). To assess the functional correctness, Matlab functions and models were used. In particular, the OFDM modulator was validated for all modes of operation by comparison with results from the LTE-compliant Matlab function *lteOFDMModulate()* (lte, b). The *lteOFDMDemodulate()* (lte, a) function does not perform synchronization or channel estimation and was only used to validate CP removal, FFT processing and active subcarrier extraction in the OFDM demodulator. Matlab scripts were produced to further validate coarse and fine synchronization, channel estimation and equalization.

4.1.1 Modulation

The general datapath structure for OFDM modulation is presented in Figure 4.1. Apart from the IFFT, the operations involved in the datapath mainly involve simple arithmetic, data selection and

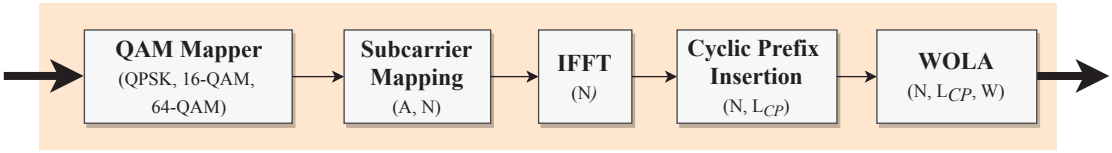


Figure 4.1: Datapath structure for OFDM baseband modulation

reordering. The first module in the OFDM modulator datapath is the QAM mapper. For a general M-ary QAM case, the module is simply implemented with an M:1 multiplexer: a $\log_2 M$-bit input signal selects a complex value out of M pre-stored constants that form the constellation. Gray mapping and average power normalization are used in the definition of the constellation point values.

After digital modulation, the *subcarrier mapping* module is responsible for mapping the A input active subcarriers to the central bins of an N-element array and zeroing the center bin - DC null subcarrier (Figure 4.2). The remaining $N-A-1$ bins correspond to null subcarriers that serve as guard bands. As the IFFT DC bin is at index 0, an IFFT shift operation is performed on the N-element array from Figure 4.2. The resulting vector is then fed to the IFFT core. Here, it is assumed that the A active subcarriers include both data and pilot subcarriers, and that the higher levels of the communication system provide them in their correct relative locations. The main constituent elements of *subcarrier mapping* are a double buffer and a control unit. The double buffer is implemented with a dual-port RAM; each of its halves stores N complex samples. This allows for simultaneous reading and writing of consecutive A-element arrays without any data conflicts: while one buffer is used for input-writing, the other is used for output-reading. The read/write access to the double buffer is managed by a control unit that receives and correctly maps the data to the correct IFFT input bin. The index mapping scheme at the control unit combines the subcarrier mapping from Figure 4.2 with the IFFT shift operation.

After IFFT processing, whose hardware implementation was discussed in Section 3.1, time-domain OFDM symbols are obtained. The next module in the datapath is responsible for *cyclic prefix (CP) insertion*. It receives a data array of size N (corresponding to a time-domain OFDM symbol) and stores it in memory. Then, the module starts to read and output the last $L_{CP}+W$

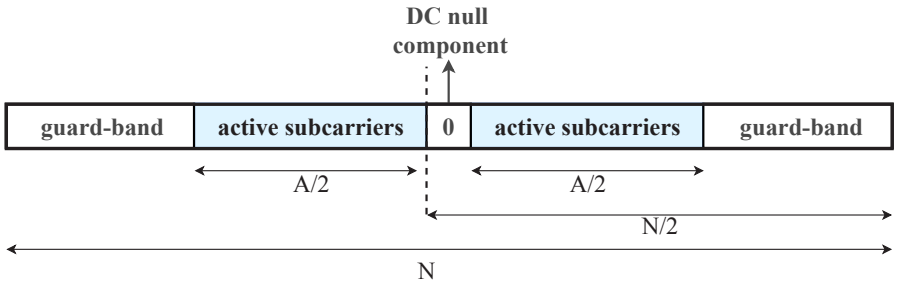
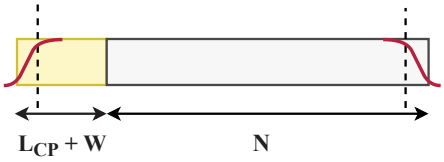
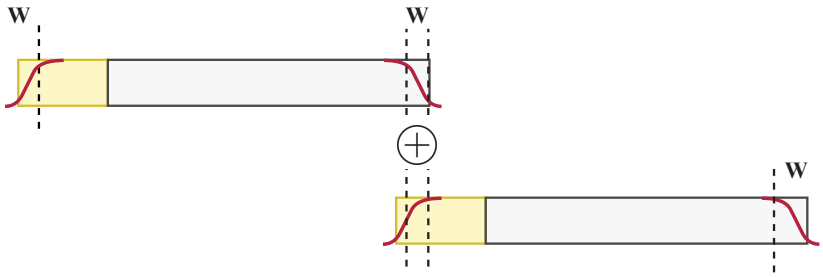


Figure 4.2: Subcarrier mapping scheme for OFDM.



(a) Windowing



(b) Overlap-and-Add

Figure 4.3: *Weighted Overlap-and-Add* (WOLA) operations.

memory positions. The cyclic prefix extension by W samples allows for the following *WOLA* operation. After outputting the last $L_{CP}+W$ memory positions, the CP insertion unit continues by reading and outputting the complete OFDM symbol from the beginning. Thus, the output of this module is an extended ODFM symbol with $N+L_{CP}+W$ complex samples. Its main hardware elements are an N-elements dual-port RAM and a control unit to command write/read memory operations.

The final module in the ODFM modulator datapath performs the *WOLA* operation (Figure 4.3). It can be divided in two stages: first, OFDM symbols are multiplied by a window - *windowing* - and then the symbol's tail is overlapped and added with next symbol's head - *overlap-and-add*. The windowing operation is implemented using two multipliers (to *window* the real and imaginary parts) and a ROM memory with pre-stored non-unitary raised-cosine window coefficients. In turn, the overlap-and-add operation is implemented with a Finite State Machine (FSM) and arrays of registers to temporarily store the OFDM symbol's head and tail- Figure 4.4.

The FSM commands the data storage and forwarding, as well as the additions involved in WOLA. The initial state is *FH* (first head) and it stores the head of the first OFDM symbol in an array of registers. This first symbol's head will be later overlapped with the tail of the last OFDM symbol processed. Afterwards, the FSM loops between three other states. The *NW* (no windowing) state corresponds to the processing of the OFDM symbol samples which are not multiplied by window coefficients. These samples are forwarded to the module's output without any additional processing. When the beginning of the symbol's tail is detected (last W symbol samples), the FSM moves to the T (tail) state. In this state, the tail samples are stored in another array of registers. After storing the symbol's tail, a new OFDM symbol arrives and the FSM switches to the *OAA* (overlap-and-add) state. The FSM remains in this state during the next W cycles, while the new symbol's head arrives. The incoming samples of the new symbol's head are aligned and added

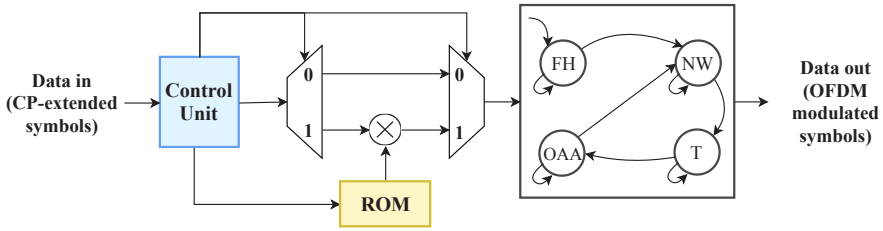


Figure 4.4: *Weighted Overlap-and-Add* architecture.

with the samples of the previous symbols' tail stored in the array of registers. The addition results are then forwarded to the output. Although the symbols at the output of the *CP insertion* have $N+L_{CP}+W$ complex samples each, the WOLA operation restores the OFDM symbol length to $N+L_{CP}$ samples.

4.1.2 Demodulation

The OFDM demodulation datapath is more complex that its modulator counterpart. Apart from CP removal and FFT processing, it involves coarse and fine synchronization, channel estimation and equalization (Figure 4.5). Considering the channel model mentioned earlier in this chapter, the received signal *r(k)* can be represented as follows:

$$r(k) = t(k-\Theta) \times e^{j\frac{2\pi\varepsilon k}{N}} + n(k), \tag{4.1}$$

where t is the transmitted signal, n is the AWGN contribution, Θ is the symbol time offset (STO), ε is the carrier frequency offset (CFO) and N is the number of subcarriers per OFDM symbol. The STO correction consists of ignoring the received signal until the instant Θ, while the CFO correction requires the multiplication of the received signal by $e^{-j\frac{2\pi\varepsilon k}{N}}$.

The coarse time/frequency synchronization aims at estimating and correcting the STO and CFO. The synchronization algorithm adopted in this dissertation is inspired by Beek's algorithm (van de Beek et al., 1997) and exploits the fact that the cyclic prefix in OFDM symbols is a repetition of the symbol's end. The cornerstone of the coarse synchronization algorithm is the correlation of the received signal with a delayed version of itself. Then, the moving sum of the correlation signal is computed over a window of L_{CP} samples. Recalling Table 4.1, one observes that, for a given mode of operation, L_{CP} is not the same in all slot symbols. For synchronization purposes, the L_{CP} for slot symbols other than the first one was considered. The moving sum is the operation used to compute estimations for STO ($\hat{\Theta}$) and CFO ($\hat{\varepsilon}$). The STO is estimated as the location of the

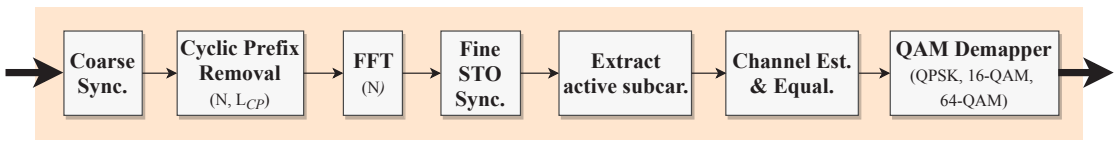


Figure 4.5: Datapath structure for OFDM baseband demodulation

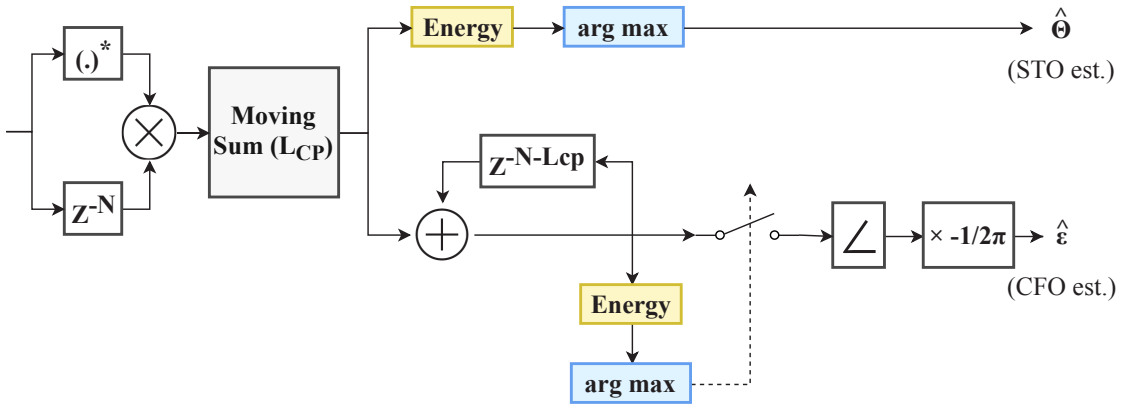


Figure 4.6: Coarse Synchronization module architecture.

maximum energy of the moving sum in the last received slot (7 OFDM symbols). The moving sum is accumulated over 7 OFDM symbols and the angle of the accumulation result at its energy peak is taken as the CFO estimation. The angle is computed by a CORDIC core configured to work in *rotation mode* and designed following the guidelines from (Meyer-Baese, 2007). Figure 4.6 shows the structure of the implemented coarse synchronization module. For sake of simplicity, all angle values are normalized by π.

To generate the complex exponential $e^{-j\frac{2\pi\hat{\varepsilon}k}{N}}$, a numerical controlled oscillator (NCO) is implemented (Figure 4.7). The NCO has a ROM-based LUT to pre-store sine and cosine values for the quadrant I - *quarter wave symmetry*. The phase of the complex exponential is incremented by the NCO step $(-2\pi\hat{\varepsilon}/N)$ every time a new input sample arrives. The accumulated phase is normalized to the interval $[0,2]\times\pi$ and the result is fed to the *quantize angle* unit. This unit is responsible for generating the address used to fetch the real and imaginary parts of the complex exponential from the ROM. As the ROM exploits the quarter wave symmetry, the *quantize angle* unit determines the quadrant for the incoming phase angle and reduces it to the first quadrant. In this case, the ROM stores 4096 32-bit phasors (16 bits for the sine and 16 bits for the cosine) that correspond to angles belonging to $[0,\frac{1}{2}]\times\pi$. Thus, the angle step between the phasors stored in ROM is $0.5\pi/4096 = 2^{-13}\pi$. The address to fetch the correct ROM phasor is obtained by dividing the reduced angle by 2^{-13} (shift left 13 times) and taking the $\log_2 4096 = 12$ most significant bits of the result. After fetching a ROM phasor, the information abut the original angle quadrant is used to set the signs of the sine and cosine values. The resulting cosine-sine pair corresponds to the real and imaginary parts of the complex exponential generated by the NCO. This complex number is then multiplied by the incoming data sample in order to correct its CFO.

Following coarse time and frequency synchronization, the cyclic prefix is removed from each OFDM symbol. The CP removal operation is controlled by 2 counters: a modulo 7 *symbol counter*, to keep track of the L_{CP} for the current OFDM symbol; and a *subcarrier counter*, to count the amount of received subcarriers for the current OFDM symbol. Based on the counter values, the *CP removal* module ignores the CP subcarriers and forwards the non-CP subcarriers to the FFT core,

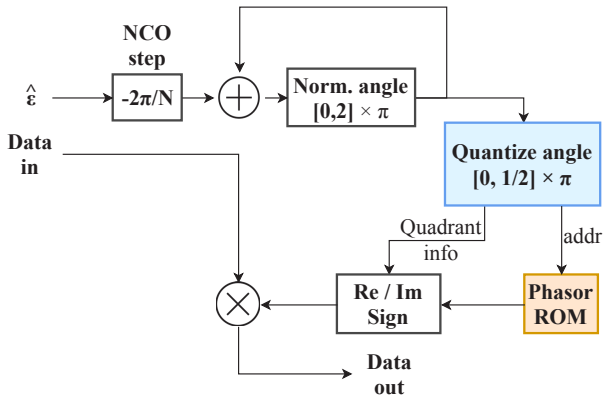


Figure 4.7: Numerically Controlled Oscillator (NCO) architecture.

whose operation and architecture was described in section 3.1. In order to avoid the overlap of OFDM symbols due to the WOLA operation and to cope with some channel delay spread, the CP removal is done halfway through the CP (lte, a). Consequently, after the FFT, a phase correction is applied to each subcarrier k through the multiplication by $e^{-j2\pi\frac{0.55L_{CP}.k}{N}}$. The complex exponential values are stored in a ROM. Another post-FFT operation is the FFT shift to revert the IFFT shift done at *subcarrier mapping* in the OFDM modulator. It is implemented using a double buffer and a $\log_2 N$-bit counter.

Some of the remaining operations in the OFDM demodulator - fine STO estimation, channel estimation and equalization - take advantage of the pilot subcarriers regularly scattered along the OFDM symbols. In LTE, pilot subcarriers are often referred as *Cell-Specific Reference Signals* (CRS) and their time-frequency location within a sub-frame (14 OFDM symbols) depends on the sub-frame index, the cell identification number (*cell-ID*) and the number of transmission antennas. Throughout this dissertation, the pattern for pilot location follows the LTE scheme considering sub-frame index 0, *cell-ID* $= 0$ and one transmission antenna (Figure 4.8). For a given mode B_X (Table 4.1), the number of pilots per sub-frame is $8N_{RB}$. The pilot values used in this work are those defined for mode B_{20}. For the remaining modes, the first $8N_{RB}$ out of 800 B_{20} pilots are used.

The synchronization algorithms based on (van de Beek et al., 1997) show a considerable fluctuation for STO estimation in the presence of long channel impulse response or low SNR. To improve the coarse STO estimation, Chang (Chang, 2008) proposes a pilot-aided fine STO estimation algorithm. As the algorithm is applied in the frequency domain, i.e. after the FFT, the effect of the residual STO ($\hat{\Theta}_f$) appears as a complex exponential $e^{-j\frac{2\pi\hat{\Theta}_f k}{N}}$ multiplied by the FFT output. Considering N as the number of subcarriers per OFDM symbol, $k1$ and $k2$ as the indexes of two consecutive pilot subcarriers and Δk as the number of data subcarriers between them $i2$ and $i1$, the metric from (Chang, 2008) to estimate the residual STO is:

$$\hat{\Theta}_f = \frac{N}{2\pi\Delta k}.\angle\sum(X_{k1}.P_{k1}^*)(X_{k2}.P_{k2}^*)^*, \tag{4.2}$$

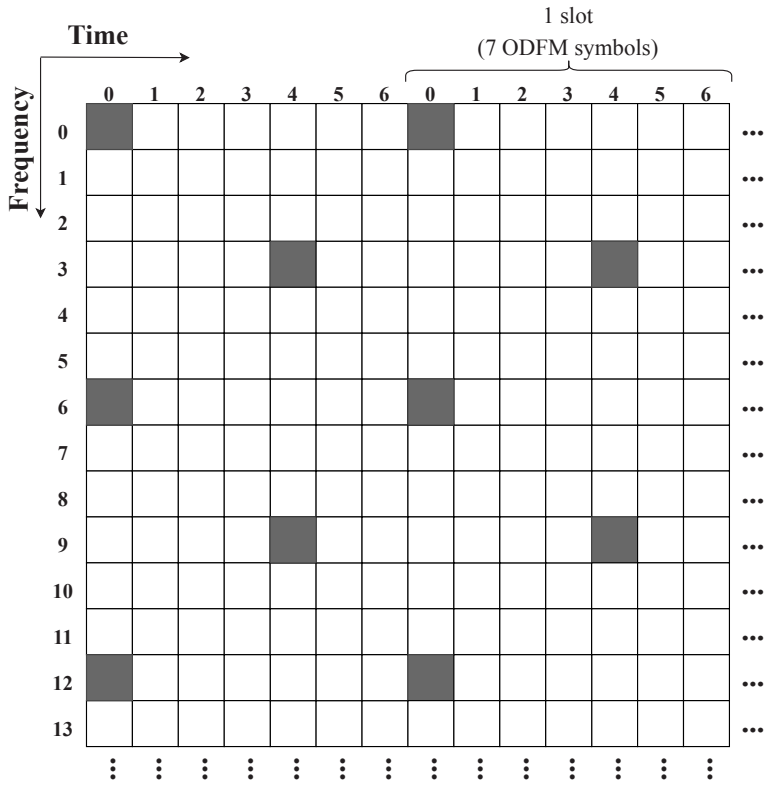


Figure 4.8: LTE pilot grid: pilot subcarriers are marked in dark grey.

where X_k is the FFT output and P_k is the expected pilot value at the k^{th} subcarrier. From Figure 4.8, one notes that $\Delta k = 5$.

The fine STO module internal structure is depicted in Figure 4.9. The base for its control unit is a subcarrier index counter ranging from 0 to $N-1$. As fine STO estimation is performed before guard-band removal, the control unit has to account for their existence when detecting pilot subcarriers. If the index of the current subcarrier is a pilot index, the control unit generates a ROM address to fetch the corresponding expected pilot. In fact, due to equation 4.2, the ROM stored values are the pilots' complex conjugate (P_k^*). Also, the metric from 4.2 works with pairs of $X \times P^*$ products. The received pilot is multiplied by the conjugate of the expected pilot and the result is either stored (*k1* indexes) or conjugated (*k2* indexes). The two consecutive $X \times P^*$ products are further multiplied and the result is accumulated over an OFDM symbol. The angle of the sum of products is calculated using a CORDIC core, as in the coarse synchronization. Here, the metric for the first OFDM is used for fine STO correction in all subsequent symbols. The fine STO correction is performed analogously to the CFO correction. The only difference is the adaptation of the NCO step to reflect the metric from 4.2.

Prior to channel estimation and equalization, the active (non-null) subcarriers are extracted from the OFDM symbols. This operation consists of removing the guard bands and DC component previously inserted during *subcarrier mapping* at the OFDM modulator. Based on the location of null subcarriers, a $\log_2 N$-bit subcarrier index counter is used to either forward active subcarriers or

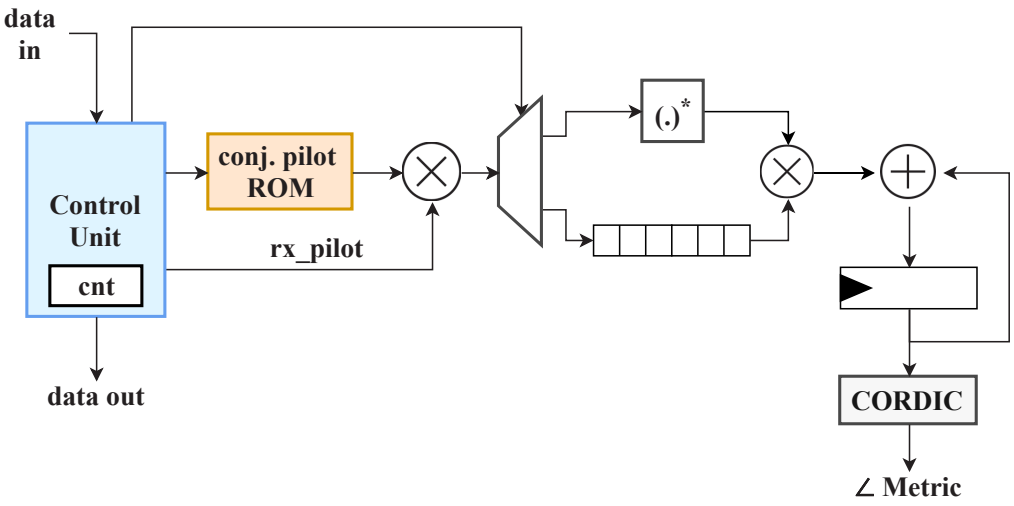


Figure 4.9: Fine Symbol Time Offset estimation (module architecture).

ignore null subcarriers. The result of this operation is a series of A-element vectors.

These A-element vectors are serially processed by the *channel estimation and equalization* module (Figure 4.10), whose operation is controlled by a 4-state FSM to identify and retrieve the received pilots. The FSM state diagram maps to the time-frequency resource grid from Figure 4.8: *S0* corresponds to receiving a A-element vector with time-index 0 (pilots starting on frequency index 0); *S1_3* corresponds to time-indexes 1, 2 and 3 (no pilots); *S4* corresponds to time-index 4 (pilots starting on frequency index 3); and *S5_6* corresponds to time-indexes 5 and 6 (no pilots). By examining the received pilots and dividing them by their expected value, the channel frequency response at the pilot subcarriers is estimated. Here, this operation is implemented through the multiplication by the reciprocal of the expected pilots, which are pre-stored in a ROM. Typically, the channel estimation at pilot subcarriers is expanded to the complete resource grid by means of interpolation algorithms. Then, frequency domain equalization is applied to the complete resource grid. This approach increases the data storage requirements and is non-casual because, to determine the channel response at a non-pilot location, at least one *future* pilot, i.e. a pilot yet to be received, is required for interpolation. In this work, the equalized channel response is that of the last received pilot, as illustrated in Figure 4.11. For the A-element vector without pilots, the channel estimation grid is the one computed for the last vector with pilots and stored in *grid RAM*: for time indexes 1, 2 and 3, the estimation vector is the one from time index 0; for time indexes 5 and 6, the estimation vector is the one from time index 4. This allows for the equalized channel response grid to be built along with channel equalization. This method is causal and considerably simplifies the hardware implementation for channel estimation and equalization.

For channel equalization, the *Zero Forcing* (ZF) algorithm is adopted. It defines the equalizer gain as the reciprocal of the estimated channel frequency response. As the reciprocal of a complex

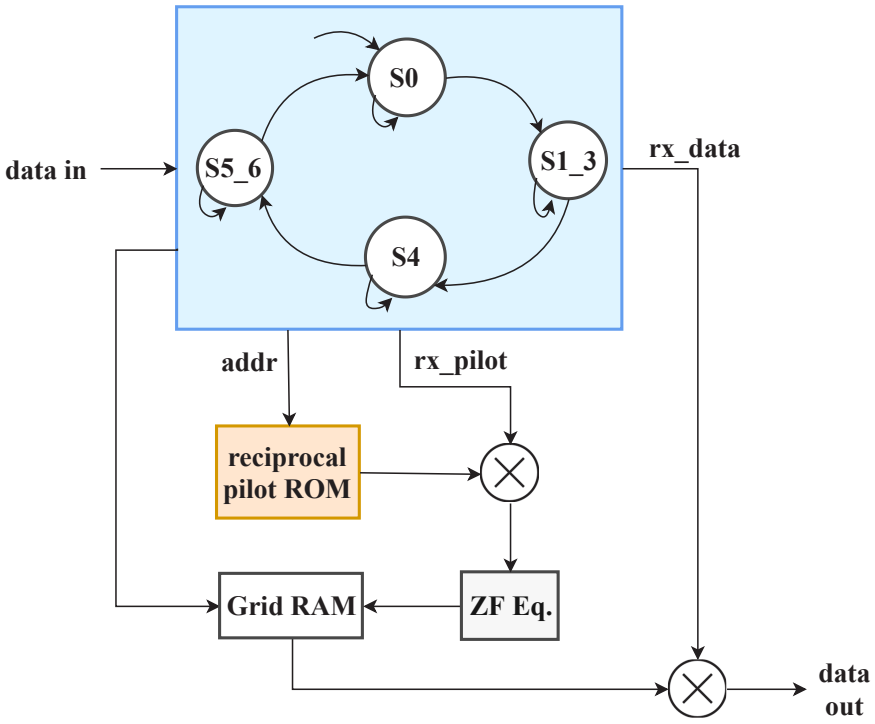


Figure 4.10: Channel Estimation and Equalization (module architecture).

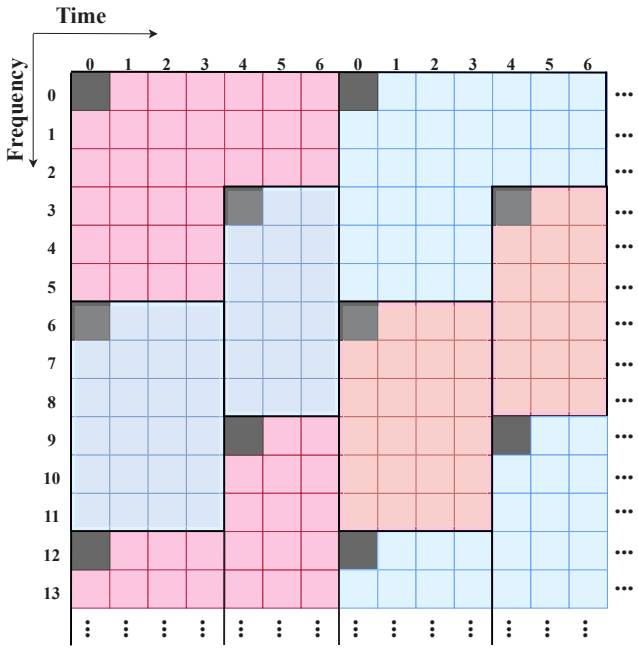


Figure 4.11: Equalized grid: blocks delimited by black rectangles have the same equalized channel value.

Table 4.2: Resource utilization for the implemented OFDM baseband modulators; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz.

Resource	Available	Mode of operation					
		$B_{1.4}$	B_3	B_5	B_{10}	B_{15}	B_{20}
Slice	13300	847	933	1059	1176	1696	1500
LUT	53200	2272	2617	2841	3371	4103	3812
FF	106400	1916	1953	2121	2183	3084	2600
BRAM	140	3	4	7	105	17	18
DSP	220	11	11	14	14	23	17

number $z = a + jb$ is defined as:

$$\frac{1}{z} = \frac{1}{a^2 + b^2}.(a - jb), \tag{4.3}$$

the ZF equalization involves two real multiplications and two real divisions. The divider blocks were implemented with the Xilinx Divider Generator and implement the Radix-2 division algorithm (Xil, 2016a). The final step in channel estimation and equalization is the multiplication of the module input data by the equalized channel response.

At the end of OFDM demodulation, QAM demapping is performed. This module uses pre-defined decision threshold values and comparators to decide upon the most likely constellation point received. Considering the constellation points as Cartesian coordinates, the decision threshold values are determined by the midpoints between neighbor points.

4.1.3 Implementation Results

Baseband processors for OFDM modulation and demodulation were implemented considering the datapath architectures just described and the numerologies from Table 4.1. The implementation results are now analyzed starting with the OFDM modulator. A first observation has to do with the QAM mapping, which is an operation independent from the channel bandwidth mode (B_X). Instead, the constellation scheme is often chosen depending on channel conditions such as the SNR. Typically, low-order constellations are used in low-SNR scenarios, resulting in smaller data rates; whereas higher-order constellations are chosen when the SNR is larger. The combined resource utilization of the three QAM mapper variants (QPSK, 16-QAM and 64-QAM) is very low: 30 slices (37 LUTs and 40 FFs). The application of run-time circuit specialization to the QAM mapper would represent a very fine granularity level for reconfiguration, with limited benefits. In turn, having QPSK, 16-QAM and 64-QAM mappers physically present on the system introduces a negligible area penalty and the constellation scheme adaptation can be done by multiplexing between the three variants.

The resource utilization for OFDM modulation in each B_X mode is presented in Table 4.2 and an additional table provides a deeper analysis on the impact each module (except for the QAM maper) has on the overall OFDM modulator consumption - Table 4.3. The resource utilization

Table 4.3: Resource variation per OFDM modulation datapath module; Post Place-and-Route results. M: maximum utilization; m: minimum utilization; Δ: utilization amplitude (maximum − minimum).

		Slice	LUT	FF	BRAM	DSP
Subcarrier Mapping	Max	51	109	94	0	0
	Min	24	54	74	0	0
	Δ	27	55	20	0	0
IFFT	Max	1187	3269	1928	12	21
	Min	556	1658	964	2	9
	Δ	631	1611	964	10	12
CP Insertion	Max	78	83	138	2	0
	Min	62	72	114	0.5	0
	Δ	16	11	24	1.5	0
WOLA	Max	401	655	750	0	2
	Min	240	475	471	0	2
	Δ	161	180	279	0	0

generally increases for higher channel bandwidth modes and it is dominated by the IFFT core. Consequently, the main trends observed for the pipelined IFFT cores (Chapter 3) also hold for the OFDM modulator. The B_{15} mode uses more slices, LUTs, FFs and DSPs. Although the DSP utilization reported in Table 3.2 is the same for 1536 and 2048-FFT, the 1536-IFFT requires additional DSPs to scale the Radix-3 butterfly results by 1/3. This is the reason for the higher DSP utilization in B_{15}. The larger number of samples processed in the B_{20} mode requires more capacity for data storage and reordering, hence its higher BRAM utilization. This applies not only to the IFFT but also for the subcarrier mapping and CP insertion modules.

The resource variation across the six modes of operation is considerable: the slice and DSP utilization of B_{15} is around twice the observed in $B_{1.4}$, whereas the BRAM utilization in B_{20} is six times larger than in $B_{1.4}$. Although all modules show considerable relative variations, the absolute variability mainly comes from the IFFT and WOLA modules, which could benefit from run-time circuit specialization.

As OFDM modulation is often simplified in the literature to IFFT processing and CP insertion, it is not straightforward to compare our designs with other published works. Still, Pham et al. (Pham et al., 2017) report resource utilization for an OFDM baseband modulator that also considers a spectrum shaping method similar to WOLA. Their design uses 1668 slices, 11 BRAMs and 15 DSPs to implement OFDM modulation for the IEEE 802.22 standard (IFFT size: 2048; $L_{CP} = 512$) on a xc6vlx240t device at $f_{clk} \approx 10$ MHz. Compared to our B_{20} mode (same IFFT size), the modulator from (Pham et al., 2017) uses less BRAMs and DSPs, but uses more slices. The multi-waveform design from (Nadal et al., 2018) (xc7z020 device) includes an OFDM modulator dimensioned for B_5 that uses 3006 FFs, 3599 LUTs as logic, 912 LUTs as RAM and 16 DSPs (post-synthesis results for $f_{clk} = 200$ MHz). Although our designs are in the same resource utilization range of (Pham

Table 4.4: Resource utilization for the implemented OFDM baseband demodulators; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz.

Resource	Available	Mode of operation					
		$B_{1.4}$	B_3	B_5	B_{10}	B_{15}	B_{20}
Slice	13300	2667	2621	2828	3000	3401	3201
LUT	53200	6668	6968	7356	7890	8822	8289
FF	106400	7202	7109	7405	7320	8404	7611
BRAM	140	13	15	22	31.5	49.5	53.5
DSP	220	45	45	48	48	51	51

et al., 2017; Nadal et al., 2018), a more detailed comparison is hard to do because of the lack of implementation details about the baseband datapath and their submodules.

Implementation results are now presented for the OFDM demodulator datapath. Similarly to the QAM mapper, the resource utilization for the QAM demapper is very low: 19 slices (12 LUTs and 18 FFs) for the QPSK, 16-QAM and 64-QAM combined. Thus, the constellation adaptation at run-time can also be done using multiplexer structures. The remaining stages of the datapath are more complex and, from Table 4.4, one observes that it requires more resources than its modulator counterpart. Like the OFDM modulator, the OFDM demodulator also shows a higher resource utilization for high channel bandwidth modes of operation. However, except for BRAMs, the resource usage variation across all modes of operation is not so pronounced as it is for the OFDM modulator.

The module-by-module analysis (Table 4.5) reveals that the resource utilization is not dominated only by the FFT-related operations: FFT, FFT shift and post-FFT phase correction. Apart from the *CP removal* and *extract active subcarriers*, the remaining modules have a significant contribution to the resource utilization of the OFDM demodulator. Another key observation is that, except for FFT-related operations and BRAM usage in *coarse synchronization*, all modules have a relatively constant resource utilization for different modes of operation. This explains the final comment of the last paragraph about the smaller variation in resource utilization for the OFDM demodulator. In fact, only the FFT internal operation is deeply affected by the parameters from Table 4.1. The operations performed in the other modules are less dependent on those parameters. For instance, ZF equalization requires a considerable amount of FFs and LUTs to compute the reciprocal of a complex number, but the nature of the operation is completely independent from the baseband parameters. Therefore, within the OFDM demodulator datapath, the FFT-related operations are those more likely to benefit from run-time circuit specialization.

In the presented design, the coarse synchronization module produces estimates and, for verification purposes, applies time and frequency correction on all received data before forwarding them to the subsequent datapath modules. As the CFO and STO estimates are produced based on the first received slot, a considerable amount of data is stored in the module. This is the reason for the high BRAM utilization associated with the coarse synchronization, especially for high channel bandwidth modes. In a practical communication system, the first data samples would be dropped

Table 4.5: Resource variation per OFDM demodulation datapath module; Post Place-and-Route results. M: maximum utilization; m: minimum utilization; Δ: utilization amplitude (maximum − minimum).

		Slice	LUT	FF	BRAM	DSP
Coarse Sync.	Max	648	1411	1260	24	13
	Min	518	1231	1219	7	13
	Δ	130	180	41	17	0
CP Removal	Max	24	69	49	0	0
	Min	20	33	49	0	0
	Δ	4	36	0	0	0
FFT, Shift and Phase Corr.	Max	1392	4074	2397	22	18
	Min	812	2211	1150	4	12
	Δ	580	1863	1247	18	6
Fine STO Sync.	Max	521	1289	945	6.5	11
	Min	482	1176	932	1	11
	Δ	39	113	13	5.5	0
Extract Active Subcarriers	Max	39	60	80	0	0
	Min	25	53	76	0	0
	Δ	14	7	4	0	0
Channel Estimation and Equalization	Max	939	1876	3657	1	9
	Min	880	1856	3657	1	9
	Δ	59	20	0	0	0

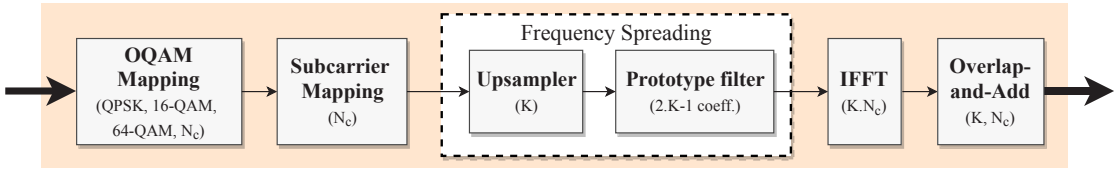


Figure 4.12: Datapath structure for FS-FBMC-OQAM baseband modulation

and only used for estimation purposes.

Besides the modulator, Pham et al. (Pham et al., 2017) also present an IEEE 802.22 OFDM demodulator that uses 6363 slices, 11 BRAMs and 33 DSPs (xc6vlx240t device, $f_{clk} \approx 10$ MHz). Our B_{20} OFDM demodulator has a higher BRAMs and DSP utilization and a lower slices utilization. The higher amount of BRAMs in our design is mainly due to the coarse synchronization module and reordering operations, such as the FFT shift. In (Pham et al., 2017), fine STO synchronization is performed in the time domain and employs multiplierless correlation between the received data and a preamble. This multiplierless correlator avoids the use of DSPs at the cost of increased slice utilization.

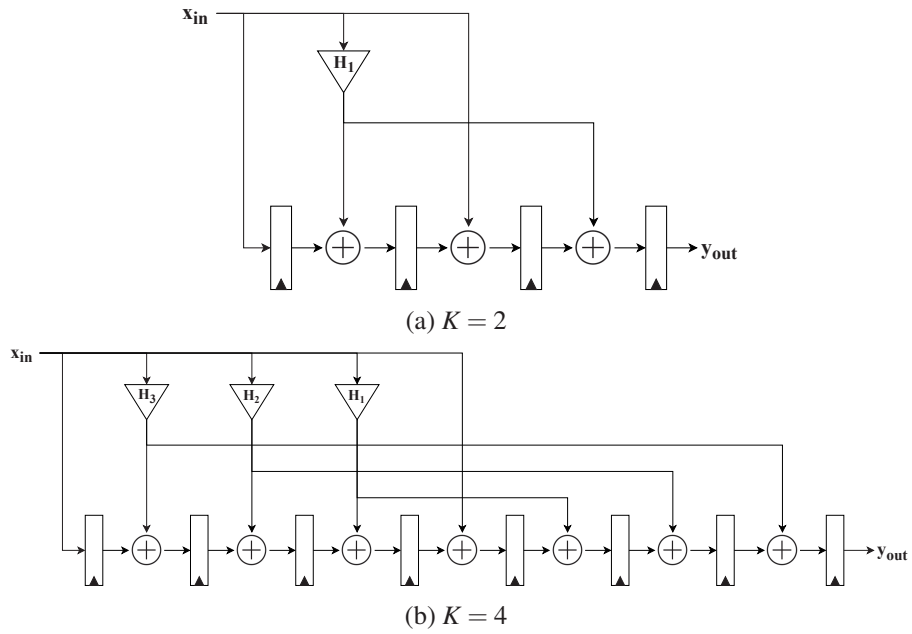
This section allowed for the evaluation of an OFDM modulator-demodulator chain and for the comparison of the algorithm and hardware implementation complexity involved on each part of the communication chain.

4.2 FBMC Baseband Processing

As stated in Section 2.1.3, this work considers the frequency spreading (FS) scheme to implement FBMC baseband modulation. Due to the lack of FBMC-based standards, values for the overlapping factor (K) and the number of subcarriers per symbol (N_c) were retrieved from published works, and four modes operation were defined: 1) $\{K = 4, N_c = 512\}$; 2) $\{K = 4, N_c = 1024\}$; 3) $\{K = 2, N_c = 1024\}$; 4)$\{K = 2, N_c = 2048\}$. In all modes of operation, 25% of N_c are used as zero-valued guard bands. The functional validation was done by comparing the output values produced by the implemented FS-FBMC modulator with the results produced by the model from (mat, a).

4.2.1 Modulation

Figure 4.12 shows the datapath structure for the FBMC modulator implemented in this work. The *OQAM Mapper* consists of two stages: first, the incoming data is QAM-modulated; then, the resulting in-phase and quadrature components are decoupled and alternately transmitted on successive subcarriers and on successive transmitted symbols (Doré et al.). For instance, if the a symbol transmits the in-phase (I) and quadrature (Q) components with the pattern *I,Q,I,Q,...*, the next symbol will transmit with the pattern *Q,I,Q,I...*. The QAM mapper is implemented as described in section 4.1.1. In turn, the I/Q decoupling is efficiently performed with an FSM that alternatively stores or outputs the I/Q components of a QAM symbol.



(a) $K = 2$

(b) $K = 4$

Figure 4.13: FIR filters architecture for pulse shaping in FBMC.

The following datapath modules are mainly characterized by K and N_c. The *guard band insertion* module places the OQAM symbols in the central bins of an N_c-elements array. The remaining subcarriers are zero-valued and represent frequency guard bands. Its operation is similar to *subcarrier mapping* in OFDM modulation. The main difference is that no DC null component is inserted. Thus, this module mainly consists of a BRAM-based double buffer and control unit to command data storage and forwarding.

The frequency spreading operation comprises *upsampling* by K and *FIR* filtering - section 2.1.3. The upsampler outputs $K-1$ zero values between two incoming I/Q samples. It uses registers to store the input data and a counter to control the number of zero values at the output. For pulse shaping, a FIR filter architecture with a transpose structure was adopted because, unlike the direct FIR model, it does not require an extra input shift register, nor a tree of pipelined adders to achieve high throughput. The number of filter coefficients lengths is odd ($2 \times K - 1$) and their values are symmetric with a single center coefficient equal to one (Table 2.1). The multiplications by the center coefficient can be ignored, as they do not affect the input value. However, the remaining coefficients imply non-trivial multiplications. The amount of non-trivial multiplications per FIR filter can be halved by exploiting coefficient symmetry. The FIR architecture for $K = 2$ and 4 are shown in Figure 4.13. As the sub-band signal to be filtered is complex-valued, two FIR filters are required to separately filter the real and imaginary parts. The algorithm and architecture for the IFFT cores used for FBMC modulation were previously described in section 3.1. Like in (mat, a), an IFFT shift operation is performed on the IFFT output blocks.

At the end of the datapath, it is necessary to *overlap-and-add* consecutive IFFT output stream blocks delayed by $N_c/2$ samples (Doré et al.). The overlap-and-add operation and the generation of transmitted symbols in FS-FBMC is described by the Matlab model from (mat, a). It considers

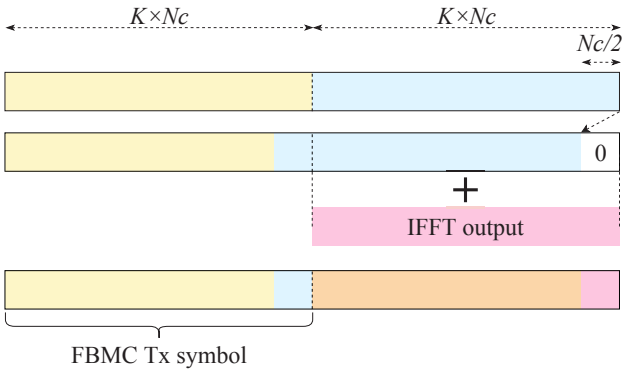
a temporary array of size $2 \times K \times N_c$, where the first half stores the current FBMC symbol to be transmitted and the second half stores the accumulation of IFFT output stream blocks. For each newly generated IFFT output block, the whole array is shifted $N_c/2$ positions and then the IFFT output block is summed to the second half of the array (Figure 4.14a). A direct mapping of this approach to a hardware implementation would be memory inefficient because it would require the use of replicated memory structures to perform two read operations per clock cycle on the temporary array (Carvalho, 2017).

Instead, the overlap-and-add (OAA) module was inspired in the work from (Bellanger, 2012). The main difference has to do with the fact that OQAM is not employed in (Bellanger, 2012) and, for the overlap-and-add operation, the consecutive IFFT output block streams are delayed by N_c. The architecture of the overlap-and-add module is shown in Figure 4.14b. To continuously accumulate consecutive IFFT output blocks delayed by $N_c/2$, a feedback shift register of $(2 \times K - 1) \times N_c/2$ samples (*Accumulator*) is used to align the previous IFFT output block with the incoming IFFT output block.

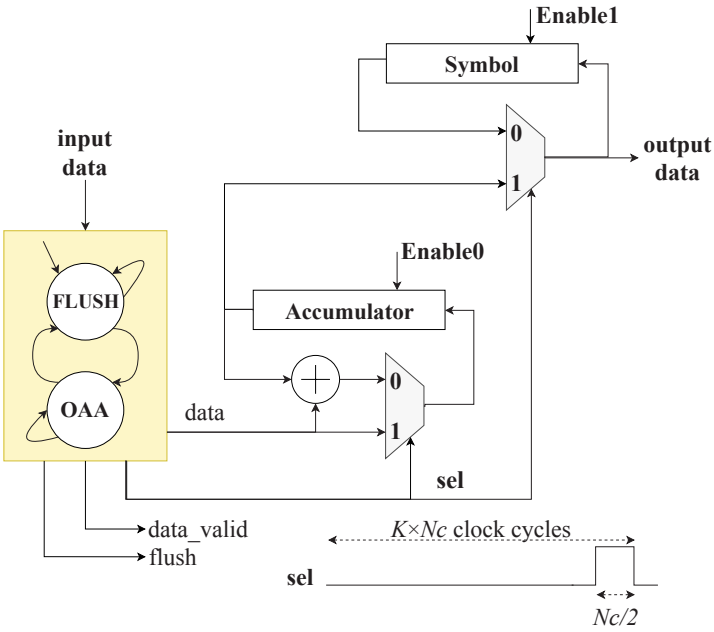
Apart from the *Accumulator*, there is another shift register of $(2 \times K - 1) \times N_c/2$ samples, denominated as *Symbol*. This shift register is used to build an FBMC output symbol according to Figure 4.14a: in a new FBMC symbol, $N_c/2$ samples come from the Accumulator shift register and the remaining samples consist of part of the previously transmitted symbol. The data forwarding within the overlap-and-add module is mainly controlled by an FSM that alternates between two states: *OAA* and *FLUSH*. When operating in the *overlap-and-add* state (*OAA*), the module receives a stream of data from the IFFT module and the FSM generates the *sel* signal with the periodic pattern shown in Figure 4.14b. This pattern is repeated every $K \times N_c$ valid input samples that arrive in the overlap-and-add module. The flag *data_valid* indicates whether the input data value is valid or not and the *flush* flag is activated during the *FLUSH* state, which initializes the shift registers with zero values. Overall, our overlap-and-add architecture requires the storage of $2 \times (2 \times K - 1) \times N_c/2 = (2 \times K - 1) \times N_c$ samples.

4.2.2 Implementation Results

As reported in section 2.1.3, the few published FPGA designs for FBMC-OQAM baseband modulators follow the PPN approach. This design choice is motivated by the lower IFFT size required, compared with the FS approach. To the best of our knowledge, the hardware design presented in the previous section is the first published FPGA-based FBMC-OQAM modulator (transmitter side) following the frequency spreading approach. After a resource utilization analysis similar to the one performed for OFDM modulation and demodulation, our FS-FBMC design is compared with the PPN-FBMC designs from (Berg et al., 2014; Nadal et al., 2016). The study of the hardware realization of baseband processing datapaths is important because it will impact the cost and energy consumption of future wireless devices. This dissertation not only contributes an FPGA-based design for FS-FBMC-OQAM, but also a comparative study between the two FBMC approaches: PPN and FS.



(a) Module operation



(b) Module architecture. Both *accumulator* and *symbol* shift registers store $(2 \times K - 1) \times N_c/2$ I/Q samples each. *Enable0* = data_valid or flush. *Enable1* = (data_valid or flush) and sel.

Figure 4.14: *Overlap-and-add* operation and architecture in the FS-FBMC modulator.

Table 4.6: Resource utilization for the implemented FBMC baseband demodulators; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz.

		Mode of operation			
Resource	Available	$K = 4$ $N_c = 512$	$K = 4$ $N_c = 1024$	$K = 2$ $N_c = 1024$	$K = 2$ $N_c = 2048$
Slice	13300	1318	1722	1217	1738
LUT	53200	3576	4878	3455	4745
FF	106400	2239	2226	2116	2103
BRAM	140	21	44	22	46
DSP	220	21	21	17	17

Table 4.7: Resource variation per FBMC modulation datapath module; Post Place-and-Route results. M: maximum utilization; m: minimum utilization; Δ: utilization amplitude (maximum − minimum).

		Slice	LUT	FF	BRAM	DSP
OQAM	Max	52	69	86	0	0
	Min	48	67	86	0	0
	Δ	4	2	0	0	0
Subcarrier Mapping	Max	28	43	83	4	0
	Min	21	36	73	1	0
	Δ	7	7	10	3	0
Frequency Spreading	Max[a]	75	235	269	0	6
	Min[b]	52	93	141	0	2
	Δ	23	142	128	0	4
IFFT and Shift	Max[c]	1495	4336	1576	34	15
	Min[d]	1018	3071	1521	16	15
	Δ	477	1521	55	18	0
Overlap-and-Add	Max	103	177	108	8	0
	Min	69	163	105	4	0
	Δ	34	14	3	4	0

[a] $K = 4$; [b] $K = 2$; [c] $K.N_c = 4096$; [d] $K.N_c = 2048$

The resource utilization for the implemented FBMC modulator is quantified in Tables 4.6 and 4.7. The modes of operation with larger values of $K \times N_c$ have a higher LUT and BRAM utilization due to the larger IFFT size and memory requirements for subcarrier mapping and overlap-and-add. The overlapping factor K also influences the amount of DSPs used for FIR filtering in the frequency spreading module. The IFFT module is the one with a more significant resource utilization. It is also the module showing a larger resource usage variation across the four modes of operation and, like in OFDM, can benefit from run-time circuit specialization. In the subcarrier mapping and overlap-and-add modules, the main source of variation is the BRAM usage. For the frequency spreading module, which includes upsampling and FIR filtering, the amount of LUTs and FFs almost doubles from $K = 2$ to $K = 4$ modes. The amount of DSPs in frequency spreading can be directly mapped to the non-trivial multiplications in FIR filters: 2×3 for $K = 4$ and 2×1 for $K = 2$ (see section 4.2 for further details). Despite its relative low resource utilization, the frequency spreading module can benefit from run-time circuit specialization, because its internal architecture is highly influenced by the baseband parameter K. For instance, the variation of K affects the value and amount of filter coefficients, as well as the corresponding multiply-and-accumulate flow graph.

When comparing the complexity associated with the PPN- and FS-FBMC modulation, it is important to understand the subtleties of each FBMC approach. In particular, the use of OQAM along with FBMC has a different impact on PPN- and FS-based approaches. OQAM processing in FS-FBMC systems was discussed in the previous section. In PPN-FBMC modulation, OQAM

Table 4.8: Resource utilization for the PPN-FBMC-OQAM modulator from (Berg et al., 2014). Baseband parameters: $K = 4$ and $N_c = 1024$.

	LUT	FF	BRAM	DSP
OQAM Proc.	561	461	1	0
1024-IFFT	3805	5131	10	13
PPN	677	577	4	16
Total	5043	6469	15	29

processing converts a block of complex QAM data into two streams with an offset of $N_c/2$ samples (one stream for the in-phase and another for the quadrature components). Then, the IFFT has to process these two streams in separate. As mentioned in section 2.1.3, the synthesis filter bank in PPN-FBMC consists of an N_c-IFFT core and a time-domain prototype filter based on a K-tap PPN. But, the use of OQAM also affects the final PPN filtering, as two PPN cores are used to filter the two N_c-sample blocks coming from the IFFT. The two PPN-filtered streams are then overlapped by $N_c/2$ samples and added to produce an output FBMC-OQAM symbol (Bellanger et al., 2010).

The FS-FBMC-OQAM modulator design from previous section is now compared to the PPN-based designs from (Berg et al., 2014; Nadal et al., 2016). In (Berg et al., 2014), the FBMC modulator design is parametrized for $K = 4$ and $N_c = 1024$, which is equivalent to our mode 2 FBMC design. It is implemented on a Xilinx Kintex-7 device and constrained to a clock frequency of 130 MHz. Although our mode 2 design is constrained to a 100 MHz clock frequency, the reported maximum clock frequency is about 125 MHz. The FBMC modulator datapath from (Berg et al., 2014) comprises 7 modules (QAM mapper, symbol padding, preamble generation, subcarriers mapping, OQAM processing, IFFT and PPN) and the resource utilization for each one is reported. This comparative analysis only considers the OQAM processing, IFFT and PPN results for three reasons. First, the QAM mapper from (Berg et al., 2014) is fed by a stream of bits from an interleaver, which makes QAM modulation more complex than in our work and also requires the symbol padding module. Second, preamble generation has a considerable impact on resource utilization, makes subcarrier mapping more complex and is not considered in our work. Third, OQAM processing, IFFT and PPN are the functional core of PPN-FBMC-OQAM modulation, while the remaining modules are not FBMC-specific. Overall, the decision to leave these modules out of the comparison makes the comparison fairer and drive the focus of the analysis to the actual trade-offs between PPN and FS-based FBMC approaches.

The resource utilization of these three modules is shown in Table 4.8. Compared to (Berg et al., 2014) and despite the larger IFFT size associated, our mode 2 design uses a similar amount of LUTs, less FFs and DSPs, but more BRAMs. This is due to the higher data storage capacity required by the 4096-IFFT core compared to a 1024-IFFT core. Moreover, the IFFT shift and overlap-and-add operations also use BRAMs to implement double-buffers and shift-registers, respectively. The DSP utilization is usually associated with non-trivial arithmetic computations. In our design and in (Berg et al., 2014), DSPs are only used in IFFT processing and prototype filtering. Although

not detailed in (Berg et al., 2014), the DSP utilization for the 1024-IFFT indicates that a pipelined FFT architecture was adopted. As expected, the pipelined 4096-IFFT core in our design uses more DSPs than the 1024-IFFT from (Berg et al., 2014). But even with a higher DSP utilization in the IFFT core, our FS-FBMC design has an overall lower DSP utilization than (Berg et al., 2014). To explain this, the DSP utilization for prototype filtering is analyzed next. In our design, the prototype filter for $K = 4$ includes two FIR filters (to filter the real and imaginary parts separately) and each filter has three DSPs to compute multiplications by non-unitary coefficients. In turn, the 4-tap PPN structure from (Berg et al., 2014) uses 16 DSPs. In general terms, the frequency spreading module in our mode 2 design uses no BRAMs and less than half the LUTs, FFs and DSPs of the PPN module in (Berg et al., 2014). Due to the lack of information about the internal structure of the PPN module from (Berg et al., 2014), it is hard to further discuss the prototype filtering in FS- and PPN-FBMC. However, the increased resource utilization in the PPN structure is probably caused by the need for two PPN cores, when using OQAM along with PPN-FBMC.

In total, our design uses about a third of the FFs reported in (Berg et al., 2014). The main discrepancy comes from the IFFT module. A possible cause for the higher FF utilization in (Berg et al., 2014) can be the over-optimization for high clock frequencies. The OQAM processing module also shows a considerable high resource utilization compared to our design. This may be due to the different nature of the OQAM processing in PPN-FBMC, where two data streams have to be generated and one of them is delayed by $N_c/2$ samples.

Nadal et al. (Nadal et al., 2016) propose a PPN-FBMC-OQAM scheme where FFT pruning is used to reduce the modulator computational complexity. The proposed datapath has 16-bit fixed-point precision and its constituent modules are: OQAM mapper, *pre-processing*, $N_c/2$-IFFT, *reorder* and two PPN cores. Post-Synthesis results for an FBMC modulator with $K = 4$ and $N_c = 512$ (mode 1 of our design) are presented (Table 4.9). The authors claim that, for a xc7z020 device, the proposed design reaches a maximum clock frequency of 220 MHz, because the critical path corresponds to "*the propagation delay related to one real multiplier DSP*". As no post-place-and-route results are provided in (Nadal et al., 2016), it is hard to confirm or validate the previous claim about the maximum clock frequency. Despite this, our mode 1 FBMC design is compared to the design from (Nadal et al., 2016). In contrast with the analysis made for (Berg et al., 2014), all datapath modules from (Nadal et al., 2016) are taken into account because they are specific to the algorithm proposed in the paper. Another limitation of the comparative analysis for (Nadal et al., 2016) has to do with the authors' design decision of not using the BRAMs available on the FPGA logic fabric to implement RAM structures. Therefore, only the LUT (used as logic), FF and DSP usage will be compared.

Our mode 1 design exhibits a lower utilization for the three types of resources. The higher LUT and FF utilization in (Nadal et al., 2016) may arise from the more demanding clock frequency constraints used for synthesis. The algorithm proposed in (Nadal et al., 2016) allows for a reduced IFFT size ($N_c/2$ instead of N_c), but the new required modules (*pre-processing*) and *reorder* are not negligible in terms of LUT and FF usage. The pre-processing unit also contributes to the DSP utilization and suggests that its computational complexity combined with $N_c/2$-IFFT would

Table 4.9: Resource utilization for the PPN-FBMC-OQAM modulator from (Nadal et al., 2016). Baseband parameters: $K = 4$ and $N_c = 512$.

	LUT	FF	DSP
OQAM mapper	300	108	0
Pre-Processing	426	106	4
256-IFFT	2534	2218	12
Reorder	420	180	0
PPN	1846	1095	16
Total	5585	3788	32

be equivalent to the N_c-IFFT complexity in the classical PPN-FBMC-OQAM approach adopted in (Berg et al., 2014). The two 4-tap PPN cores combine for 16 DSPs. Thus, the amount of DSPs per PPN core is superior to the DSPs used in our frequency spreading module. This suggests that, even when OQAM is not used, the prototype filtering in PPN-FBMC may have a higher resource complexity than in FS-FBMC.

Despite the limitations on the comparison to PPN-FBMC designs, the results allow for the following conclusion:

The higher computational complexity associated to FS-FBMC-OQAM modulation does not translate into a higher resource utilization compared to equivalent PPN-based designs in FPGA implementations.

This conclusion is driven by two observations. First, the resource utilization for commonly adopted FFT architectures does not scale linearly with the FFT/IFFT size, which mitigates the overhead of larger IFFT sizes in FS-FBMC. Second, when OQAM is employed, the PPN-FBMC approach requires two PPN cores, which will double the hardware complexity of time-domain prototype filtering. In comparison, the frequency-domain prototype filtering in FS-FBMC is shown to be less resource demanding. Regarding the overlap-and-add operation at the end of the FS-FBMC datapath, its impact is more significant in terms of storage requirements (BRAMs). The FS-FBMC modulation approach is more closely related to the principle of OFDM (Bellanger, 2012) and the general MCM model presented in Figure 2.1. Therefore, by simplifying the FBMC concept without any hardware complexity and resource utilization penalty, FS-FBMC becomes a more attractive scheme for the hardware implementation of efficient FBMC systems.

4.3 UFMC Baseband Processing

To ease waveform coexistence, UFMC numerologies are usually aligned with LTE channelization (Jafri et al., 2018). For instance, the PRB size normally used in UFMC published works is 12 subcarriers and the filter length L is equal to the LTE cyclic prefix length plus one: $L = L_{CP} + 1$. Similar to (mat, b; Jafri et al., 2018; Nadal et al., 2018), Dolph-Chebyshev filters are used in the presented implementation to filter each UFMC subband. It is assumed that the utilization of UFMC

Table 4.10: Baseband parameters for UFMC modulation. N: number of available subcarriers; $\frac{N}{N'}$: upsampling factor; L: filter length; filter type: Dolph-Chebyshev (60-dB side lobe attenuation)

Mode of operation	1	2
N	512	1024
PRB size	12	12
Active PRBs	3	3
IFFT size (N')	64	64
$\frac{N}{N'}$	8	16
L	37	73

occurs in DSA scenarios such as the ones described in (Kaltenberger et al., 2015), where few (e.g. 1 to 3) active PRBs are used for UFMC transmission. In this dissertation, the number of PRBs was set to three. The two UFMC modes of operation defined are presented in Table 4.10 and are based on B_5 and B_{10} modes the from Table 4.1. Following the approach from (Knopp et al., 2016) described in section 2.1.2, the general datapath architecture for UFMC modulation is shown in Figure 4.15. The functional correctness was validated by comparing the results produced by the implemented design with the results from (mat, b).

4.3.1 Modulation

The UFMC modulator has three processing branches, one to process each active PRB. These branches share the same architecture and start with QAM mapping of the incoming data. The QAM mapper is equal to the one used in the OFDM and FBMC datapaths. The *subcarrier mapping* module maps the 12 PRB subcarriers to the central bins of an array with N' (64) elements and zeroes the remaining $N' - 12$ elements. It follows the same implementation approach of subcarrier mapping in FBMC: a double buffer of $2 \times N'$ elements and read/write control engines. The IFFT module used in each processing branch was discussed in section 3.2. The *upsampler* architecture

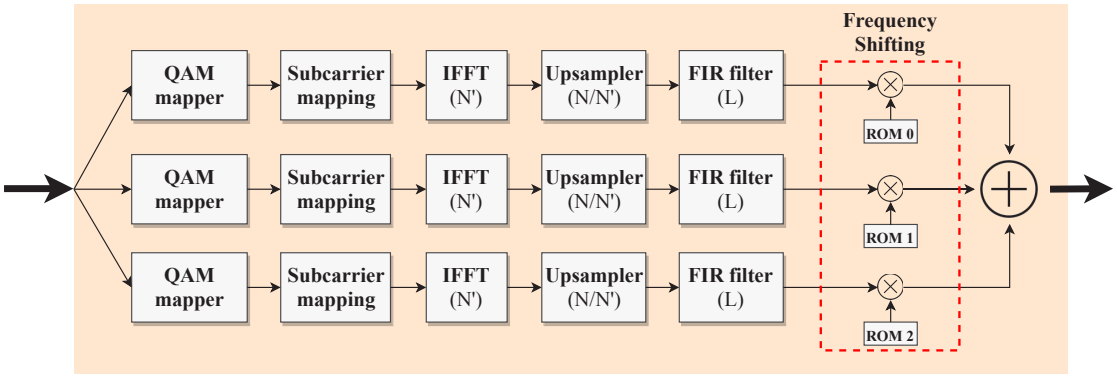


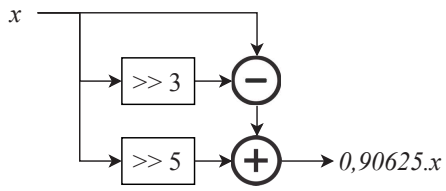
Figure 4.15: Datapath structure for UFMC baseband modulation.

and operation is similar to the one used for frequency spreading in FBMC (section 4.2). Here, the number of zeros between consecutive IFFT output samples is $\frac{N}{N'} - 1$.

Dolph-Chebyshev FIR filters with a transpose structure are used for bandpass sub-band filtering. The observations made in section 4.2 about FIR coefficients symmetry also hold in this case: there is an odd number of symmetric coefficients and the center coefficient is equal to one. However, the higher FIR order used in UFMC modulation requires further discussion. Considering an L-order FIR filter, $L-1$ coefficients imply non-trivial multiplications that can be halved due to coefficient symmetry ($\frac{L-1}{2}$). As each processing branch requires two FIR filters - for the real and imaginary parts - there are $L-1$ non-trivial multiplications per branch.

In Xilinx FPGAs, non-trivial multiplications can be efficiently performed by DSP blocks. These blocks are embedded into the logic fabric in a column arrangement. Cost-optimized devices have a smaller amount of DSP blocks and their utilization should be carefully considered. For instance, the xc7z020 device has 220 DSP blocks available. Considering the modes of operation from 4.10, the overall amount of non-trivial multiplications for FIR filtering ($3 \times (L-1)$) is 108 for Mode 1 and 216 for Mode 2. This represents a high DSP utilization and its sparse distribution within the logic fabric degrades the scalability of the UFMC modulator. Moreover, the place-and-route tasks performed by EDA tools become more challenging and likely to affect the overall timing closure.

To alleviate the DSP utilization, a multiplier-less architecture for FIR filters was adopted. The FIR coefficients are represented in Q1.5 format, using the Canonic Signed Digit (CSD) system with minimum non-zero bits. Then, non-trivial multiplications are substituted by shift-and-add graphs. For instance, considering the multiplication by a coefficient equal to 0.90625, we have:



$$\begin{aligned} 0.90625_{10} &= 0.11101_2 \\ &= 1.00\bar{1}01_{2,CSD} \\ &= (1-2^{-3}+2^{-5})_{10}. \end{aligned} \tag{4.4}$$

This strategy eliminates the use of DSP blocks in FIR filters, but increases slice utilization. Yet, slices are the most numerous resource type in the FPGA logic fabric (13,300 slices in the xc7z020 device), which makes this approach viable.

After FIR filtering, each sub-band signal is shifted to the corresponding frequency band. The *frequency shift* module for each branch has a ROM memory to store the complex exponential values and a complex multiplier similar to the one used in the FFT/IFFT modules (chapter 3). Finally, the filtered sub-band responses are summed to create the UFMC signal to be transmitted.

4.3.2 Implementation Results

Table 4.11 presents the resource utilization for the two modes of operation of the UFMC modulator. Considering the three waveform modulations studied in this work, UFMC modulation has the highest slice, LUT and FF utilization. From Table 4.12, one observes that the resource consumption for each UFMC branch is dominated by the FIR filtering module. The resource usage of the IFFT core is not so significant as in OFDM or FBMC modulation, due to the adoption of the

Table 4.11: Resource utilization for the implemented UFMC baseband demodulators; Post Place-and-Route results; Device: xc7z020; f_{clk} = 100 MHz.

Resource	Available	Mode 1		Mode 2	
		Total	per PRB	Total	per PRB
Slice	13300	2383	806	3256	1094
LUT	53200	8077	2627	11654	3816
FF	106400	6286	1972	9919	3171
BRAM	140	12	4	12	4
DSP	220	18	6	18	6

memory-based IFFT architecture (Section 3.2). Another observation has to do with the irrelevant variation of the resources used by the datapath modules other than the FIR filter structure. In fact, the IFFT and subcarrier mapping modules used in mode 1 and 2 have exactly the same RTL description. The *upsampler* and *frequency shift* modules are also very similar. The main differences are the ROM contents and size in the frequency shift, as well as the control counter range in both upsampler and frequency shift. Thus, the resource utilization variation between modes 1 and 2 is almost exclusively due to the FIR filter structure. The multiplierless design (section 4.3) is the reason for the absence of DSPs and the increased slice, LUT and FF utilization in the FIR filters. In particular, the number of FFs in the FIR filtering module scales almost linearly with the filter order, that is, the FF usage almost doubles from mode 1 (filter order = 37) to mode 2 (filter order = 73). For this reason and the fact that the parameter L has a considerable impact on the internal module architecture, the FIR filter structure can benefit from run-time circuit specialization in multi-mode environments.

Compared to the UFMC modulators from (Nadal et al., 2018; Jafri et al., 2018), our design uses more LUTs, FFs, and BRAMs. However, a careful analysis of the numerologies is required. The modulator from (Nadal et al., 2018) is dimensioned for a numerology similar to our mode 1, but considers only one PRB. The reported post-synthesis resource utilization is 4688 LUTs (as logic), 1257 LUTs (as RAM), 3073 FFs and 20 DSPs, considering a xc7z020 device and $f_{clk} = 200$ MHz. As Nadal et al. (2018) use LUTs instead of BRAMs to implement RAM structures, a direct LUT usage comparison is hard to do. Still, a single PRB branch in our UFMC modulator uses 36% FFs and 70% DSPs less than the UFMC modulator from (Nadal et al., 2018). The UFMC modulator from (Jafri et al., 2018) has a numerology similar to mode 2 in our work, but considers 25 PRBs serially processed in a double-buffer fashion. Thus, the architecture consists of the computational resources to process a single PRB and control structures to allow for the continuous processing of several PRBs. The reported resource utilization (xc7v2000t device, $f_{clk} = 364$ MHz) is 1133 LUTs, 910 FFs, 3 BRAMs and 64 DSPs. As previously stated, the higher LUT and FF utilization per PRB in our design is mainly due to the shift-and-add operations on the bandpass FIR filters. On the other hand, the DSP block utilization per PRB is around ten times lower than in (Jafri et al., 2018). An additional analysis can be made for the processing latency. In our design, the UFMC datapath latency is equal to the latency of each PRB processing branch, because all branches are

Table 4.12: Resource variation per module in each UFMC modulation branch; Post Place-and-Route results. M: maximum utilization; m: minimum utilization; Δ: utilization amplitude (maximum − minimum).

		Slice	LUT	FF	BRAM	DSP
Subcarrier Mapping	Max	25	44	57	1	0
	Min	20	38	57	1	0
	Δ	5	6	0	0	0
IFFT	Max	181	536	503	2	3
	Min	166	518	503	2	3
	Δ	15	18	0	0	0
Upsampler	Max	23	42	51	0	0
	Min	16	41	50	0	0
	Δ	7	1	0	0	0
Filtering	Max	810	3001	2434	0	0
	Min	500	1798	1238	0	0
	Δ	310	1203	1196	0	0
Frequency Shifter	Max	82	213	126	1	3
	Min	77	212	124	1	3
	Δ	5	1	2	0	0

processed in parallel. For both modes of operation, the latency per PRB is 421 clock cycles, which is lower than the 516 clock cycles reported in (Jafri et al., 2018).

4.4 Summary

This chapter presented FPGA-based designs for four baseband processing datapaths: OFDM modulation and demodulation, FBMC and UFMC modulation. After describing their architecture and operation, post Place-and-Route resource utilization for different baseband numerologies was studied. The OFDM modulator exhibits considerable resource usage variation across different numerologies. The IFFT is the most resource-demanding module in the datapath and also the one whose resource usage varies the most with different numerologies. Besides the IFFT, the WOLA module can also benefit from run-time circuit specialization. The OFDM demodulator is more complex and resource-demanding than its modulator counterpart. This was expected due to additional complex operations such as coarse/fine synchronization, channel estimation and equalization. Despite its overall higher resource utilization, the variation across different numerologies is not so evident in OFDM demodulation and dynamic circuit specialization would only benefit the FFT-related operations.

The first published FS-FBMC-OQAM baseband modulator was presented in this chapter. It was further compared to published PPN-based designs. As in the OFDM modulator, the IFFT is the most resource-demanding module and appears as an opportunity for circuit specialization at

run-time. Likewise, the frequency spreading module can also be specialized as a function of the overlapping factor K. The comparison with PPN-based designs suggests that, despite the higher computational complexity, FS-FBMC-OQAM modulation uses less resources than an equivalent PPN-FBMC design, making FS-FBMC a convenient scheme for the implementation of FBMC systems on FPGAs.

The proposed UFMC modulator considers a multiplierless filter design to reduce the DSP utilization in small-scale, cost-effective FPGA devices. The filtering structure represents the bulk of resource utilization in the UFMC modulator and its considerable variation for different numerologies makes its suitable for run-time circuit specialization.

The baseband datapaths presented in this chapter are the object for the application of run-time reconfigurability in the context of multi-mode communications. The presented study on resource utilization for different numerologies allowed the identification of opportunities for dynamic reconfiguration. Some of them will be explored and studied in the next chapter.

Chapter 5

Dynamically Reconfigurable Architectures for Baseband Processors

The goal of this chapter is aligned with the main objective of this dissertation: *the investigation and design of dynamically reconfigurable baseband processing architectures for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation*. To do so, two types of run-time reconfiguration (RTR) techniques will be applied to the baseband processing datapaths studied in the previous chapter. The first type of RTR is the *specialization for computation* at run-time (section 5.1). It allows for the adaptation of the datapath architecture or functioning according to the mode of operation demanded by the communication environment. Here, *mode of operation* refers to a set of baseband parameters such as FFT/IFFT size, CP length, overlapping factor, sub-band filter length, etc. The second type of RTR is the *specialization of performance* at run-time (section 5.2) and consist of adapting the processing throughput of the baseband datapath in response to the variation of on sampling rate or bandwidth requirements. Here, it is not intended to modify the baseband processor datapath, but instead to have a performance-scalable architecture able to provide a satisfactory range of processing throughput modes at relatively low clock frequencies. Finally, the two forms of RTR are combined to produce an architecture for a reconfigurable baseband modulator suitable for multi-mode transmission and spectrum aggregation communication scenarios (section 5.3).

5.1 The Specialization of Computation at Run-Time

The hypothesis formulated in Chapter 1 assumes that FPGA-based Dynamic Partial Reconfiguration (DPR), an enabler for specialization of computation at run-time, provides a convenient architectural concept to design flexible, efficient and upgradeable baseband processors for multi-mode communications. This section presents three studies on the application of DPR in baseband processing datapaths. The first one addresses the design of a dynamically reconfigurable OFDM modulator and compares two datapath partition strategies. The purpose is to acquire detailed knowledge on the trade-offs involved in the definition and dimensioning of reconfigurable partitions. The second

study targets a dynamically reconfigurable FS-FBMC modulator. Two design approaches with different granularity levels for reconfiguration are presented and compared. The results of the first two studies focus on the amount of resources used or reserved by the reconfigurable datapaths, as well as the size of the partial bitstreams involved in DPR. As the specifications of each particular case involve different trade-offs, it is not intended to provide general or systematic conclusions about the best way to implement dynamically reconfigurable baseband processors, but instead to expose the challenges and considerations related to the design of this kind of systems.

The third and last study of this section coincides with one of the contributions of this dissertation: the evaluation and comparison between a DPR-based and a static multi-mode design for a reconfigurable baseband modulator. The metrics considered for analysis are resource utilization, performance, functional density and power consumption. Additionally, this study also contributes a quantitative analysis of the DPR impact in terms of reconfiguration latency and energy overhead. The obtained results allow for a discussion of the viability of applying DPR in baseband processing for modern communications.

5.1.1 Dynamically Reconfigurable OFDM baseband modulator

The design of a dynamically reconfigurable OFDM modulator is motivated by the considerable resource usage variation across different modes of operations, as reported in section 4.1.3. Recalling the OFDM parameters from Table 4.1, only the parameter W presents repeated values in more than one mode of operation. This means that switching the mode of operation requires the adaptation of all datapath modules. Although only the IFFT and WOLA modules were identified as good opportunities for DPR, the overhead of reconfiguring the whole datapath is relatively low and allows for the circuit specialization of each module. The QAM mapper is excluded from this analysis due to its simplicity and the QAM constellation adaptation is performed using mux-demux structures. Although DPR is to be applied to the whole modulator datapath, there are different strategies to define reconfigurable partitions (RPs). Two partition strategies are studied in this subsection and, for each one, a dynamically reconfigurable OFDM modulator is implemented on a xc7z020 device.

The first partition strategy considers a single RP to implement the OFDM modulator datapath: subcarrier mapping, IFFT, CP insertion and WOLA. The goal of this approach is the global place-and-route optimization of the datapath inside the RP. This may lead to a reduced overall amount of RP reserved resources and partial bitstream size.

The second strategy attempts to identify hardware blocks common to all modes of operation that do not need to be reconfigured. The common hardware blocks are placed in the system's static part, thus reducing the amount of resources that have to be reserved for DPR. A careful analysis of the IFFT sizes in Table 4.1 reveals that all of them are multiples of 64. For the pipelined FFT architecture, this means that the hardware to implement a 64-FFT core is necessary in all modes of operation and can remain in the system's static part. The resource utilization for the 64-FFT is not negligible (1280 LUTs, 456 FFs, 0.5 BRAMs and 6 DSPs) and, for instance, accounts for more LUTs and DSPs than the subcarrier mapping, CP insertion and WOLA modules together. The remaining datapath logic varies from one mode of operation to another and will

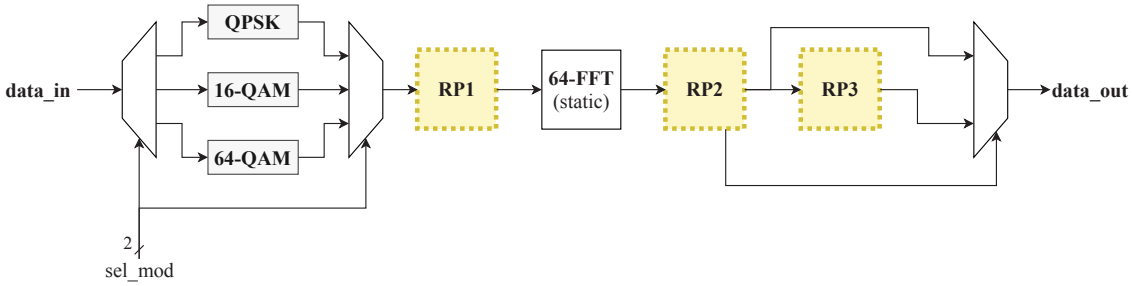


Figure 5.1: Dynamically reconfigurable OFDM baseband modulation with multiple RPs.

be reconfigured through DPR. The 64-FFT splits the OFDM modulator datapath in two halves. The first half implements the subcarrier mapping and IFFT-related operations before the 64-FFT (*pre-64-FFT operations*). The second half accommodates the IFFT-related operations after the 64-FFT (*post-64-FFT operations*), CP insertion and WOLA.

Figure 5.1 illustrates the multiple RP approach for the dynamically reconfigurable OFDM modulator. A reconfigurable partition (*RP1*) is defined for the first datapath half: subcarrier mapping and pre-64-FFT operations. The minimum resource requirements for RP1 are imposed by the B_{20} mode: 1468 LUTs, 504 FFs, 11 BRAMs and 6 DSPs. A second partition (*RP2*) is created to implement post-64-FFT operations, CP insertion and WOLA for all modes of operation except B_{15}. Regarding B_{15}, only the 1536-IFFT operations related to bit-reversal reordering and the Radix-2 stage are placed in RP2. The remaining B_{15} operations (Radix-3 stage, CP insertion and WOLA) are implemented in a third partition (*RP3*). The reason behind this design decision is the higher complexity of the B_{15} datapath after the 64-FFT. The datapath operations after the 64-FFT require 1082 LUTs, 1215 FFs, 6.5 BRAMs and 5 DSPs in the B_{20} mode and 2076 LUTs, 1848 FFs, 7 BRAMs and 14 DSPs in the B_{15} mode. Expanding RP2 to also implement the later stages of B_{15} would result in RP2 under-utilization for five out of six modes of operation. Thus, RP2 was dimensioned to implement post-64-FFT operation, CP insertion and WOLA in B_{20}, while RP3 maximum resource utilization is dictated by Radix-3 stage, CP insertion and WOLA in B_{15} (1728 LUTs, 1546 FFs, 4.5 BRAMs and 11 DSPs). Considering a broader communication system , RP3 could be reused for other system tasks when the OFDM is not operating in the B_{15} mode. As the datapath output can come from RP2 or RP3, there is a multiplexer in the static part to correctly forward the results to the OFDM modulator output.

Table 5.1 provides a comparison between the two partition strategies in terms of resource utilization and partial bitstream sizes. While the resource utilization in the single RP design consists of the resources reserved for the RP, in the multiple RP design it consists of the resources reserved for all RPs plus the resources used by the 64-FFT. The multiple RP design keeps the 64-FFT as static logic and, thus, has less datapath logic to implement on the RPs. Despite this, it reserves more resources than the single RP design. An immediate reason for this observation is the fact that breaking the datapath in two or more RPS prevents global optimization on the whole datapath. Another reason has to do with the distribution of elements (CLBs, BRAMs and DSPs) on the FPGA architecture, as well as the constraints on RP dimensioning imposed by vendor tools.

Table 5.1: Resource utilization and partial bitstream sizes for the dynamically reconfigurable OFDM baseband modulator. The numbers in parentheses are the maximum resource utilization within the corresponding RP.

	Multiple RPs					Single RP
	64-FFT	RP1	RP2	RP3	Total	
Slice	426	420 (99 %)	400 (96 %)	600 (98 %)	1846	1400 (95 %)
LUT	1280	1680 (87 %)	1600 (70 %)	2400 (77 %)	6960	5600 (73 %)
FF	456	3360 (18 %)	3200 (45 %)	4800 (35 %)	11816	11200 (28 %)
BRAM	0.5	12 (92 %)	10 (70 %)	10 (50 %)	32.5	20 (90 %)
DSP	6	12 (50 %)	20 (25 %)	20 (55 %)	58	40 (58 %)
Bitstream size (KB)	-	235	131	160	526	346

As an example, Figure 5.2 shows the shape and composition of RP1. Xilinx prohibits the positioning of RP vertical boundaries on interconnect columns Xil (2015b). The left and right edges of an RP must lay on CLB-CLB, CLB-BRAM or CLB-DSP borders. That said, the horizontal shrinking of RP1 would let a BRAM (left-most) or DSP (right-most) column outside the partition. Consequently, RP1 would not have enough BRAMs or DSPs to implement the first half of the B_{20} datapath (11 BRAMs and 6 DSPs required). The maximum LUT and BRAM utilization in RP1 is already 87% and 92%, respectively. Reducing the height of RP1 would hamper the floorplaning and routing within RP1. These constraints also condition RP2 and RP3, leading to a higher resource overhead in the multiple RP approach. Despite this, all RPs show a relatively high maximum slice, BRAM and DSP utilization.

Regarding partial bitstream size, the single RP approach is also more advantageous. Every time the OFDM modulator is reconfigured, a 346 kB bitstream is loaded into the FPGA configuration memory. For the multiple RP design, there are two cases: 366 kB to reconfigure RP1 and RP2 for all modes except B_{15}; or 526 kB to reconfigure all three RPs (B_{15} mode). For a fairer comparison, all partial bitstream sizes correspond to non-compressed bitstreams. Apart from reserving more resources, the reconfiguration of the multiple RP design involves the writing of a larger amount of data to the FPGA configuration memory, incurring in larger reconfiguration latency. This study case illustrates some trade-offs on datapath partition strategies and, for the proposed dynamically reconfigurable OFDM modulator, a single RP design results in reduced resource utilization, smaller partial bitstream sizes and therefore smaller reconfiguration times.

5.1.2 Dynamically Reconfigurable FS-FBMC baseband modulator

In the context of reconfigurable baseband processors for multi-mode communications, FS-FBMC systems allow the adjustment of the number of subcarriers per symbol (N_c) by varying the overlapping factor K and without reconfiguring the $K \times N_c$-IFFT block (Bellanger, 2012). However,

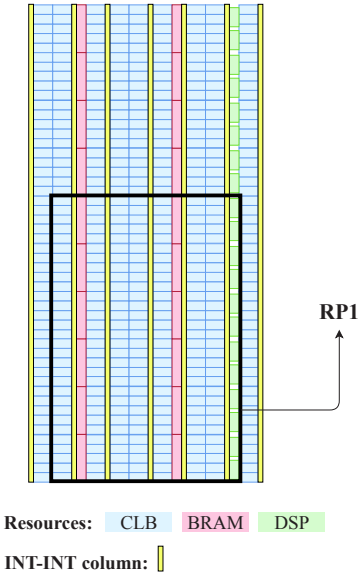


Figure 5.2: Floorplaning of RP1 in the multiple RP OFDM modulator design. *INT-INT*: interconnect column.

in some scenarios it may be desirable to maintain N_c and adjust K to control the filter bank interference levels (Ihalainen et al., 2010). This later scenario affects the $K \times N_c$ product and requires the reconfiguration of the IFFT core. The FBMC modes of operation defined in section 4.2 are reproduced in Table 5.2. There are some commonalities between different modes of operation. For example, mode 1 shares the same IFFT size with mode 3 and the same K factor with mode 2. This subsection presents two designs for a dynamically reconfigurable FS-FBMC baseband modulator, exploring different granularity levels for reconfiguration. Both designs were implemented on a xc7z020 device.

The first design approach (*single RP*) implements the FS-FBMC modulator datapath in a single reconfigurable partition aiming at increasing the place-and-route global efficiency inside the RP. To accommodate the four FBMC modes of operation, RP1 must contain at least 4878 LUTs, 2239 FFs, 46 BRAMs and 21 DSPs (Table 4.6). This coarse granularity level for reconfiguration requires that the whole datapath is reconfigured, even when the parameter variation does not affect all datapath modules.

The second design approach, depicted in Figure 5.3, takes into account that the impact of

Table 5.2: FBMC modes of operation.

Mode of operation	K	Nc	IFFT size
1	4	512	2048
2	4	1024	4096
3	2	1024	2048
4	2	2048	4096

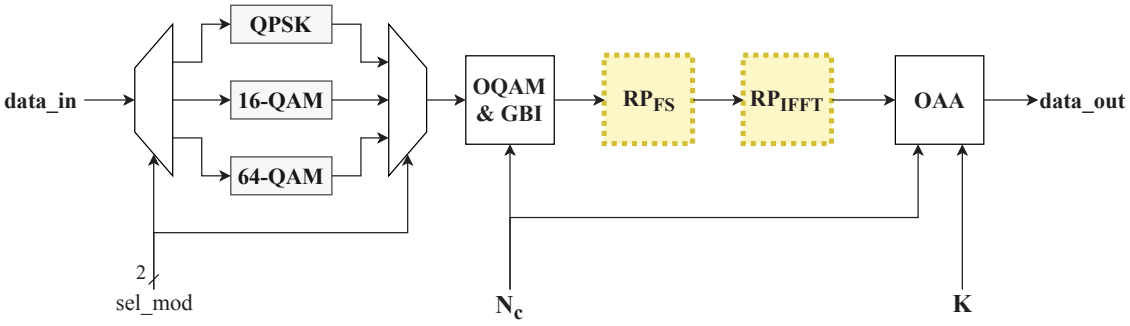


Figure 5.3: Dynamically reconfigurable FS-FBMC baseband modulation: *hybrid design*.

baseband parameter variation is not the same in every datapath module. This way, the resource utilization results from section 4.2.2 are the base to decide on how each module in the FS-FBMC modulator should be reconfigured. In section 4.2.2 it was noted that the frequency spreading and IFFT modules can benefit from run-time circuit specialization. Based on that, DPR is applied to these modules and two reconfigurable partitions are created: RP_{FS} for the frequency spreading module and RP_{IFFT} for the IFFT module. Both OQAM mapper and subcarrier mapping modules depend only on N_c. In the OQAM mapper, N_c is used to control the alternation between *I-Q-I-Q* and *Q-I-Q-I* patterns on consecutive multicarrier symbols, while in the guard band insertion, N_c affects the double buffer size and read/writing addressing schemes. In these modules, the impact of N_c does not represent significant circuitry changes or resource usage variation. Consequently, the OQAM mapper and subcarrier mapping modules are implemented via a *static multi-mode* approach: the modules are dimensioned for the most resource-demanding scenario and their operation is tuned by setting parameter values. The same applies to the overlap-and-add module, where K and N_c affect the depth of the two shift registers and the control of the FSM transitions. As it combines DPR with static multi-mode modules, this approach to design the FS-FBMC modulator is designated as *hybrid design*. With a finer granularity level for reconfiguration (module-by-module), the hybrid design approach can explore the commonalities between FBMC modes of operation pointed out previously. For instance, to reconfigure the FS-FBMC modulator from mode 2 to mode 4, it is only necessary to load a partial bitstream to adapt the circuit inside RP_{FS} and change the value of parameter N_c. The IFFT module, which is the most resource-demanding module, remains unchanged.

Table 5.3 shows resource utilization and partial bitstream size results for both dynamically reconfigurable FS-FBMC designs. The single RP design reserves more resources (except for BRAMs) than the resources used and reserved in the hybrid design. Analyzing the RP maximum resource utilization in the single RP design, one observes that the BRAM requirement imposed by FBMC mode 4 (46 BRAMs) leads to an increased RP reserved area and lower percentage of slice utilization.

The adaptation of the static multi-mode modules consists of changing the values of K and N_c on the respective input ports. This operation occurs in few clock cycles and its reconfiguration latency is negligible. For the DPR-based modules, partial bitstreams have to be written into the

Table 5.3: Resource utilization and partial bitstream sizes for the dynamically reconfigurable FS-FBMC baseband modulator. The numbers in parentheses are the maximum resource utilization within the corresponding RP.

	Hybrid design					Single RP design
	OQAM and GBI	RP_{FS}	RP_{IFFT}	OAA	Total	
Slice	130	120 (53 %)	1360 (91 %)	152	1762	1840 (77 %)
LUT	325	480 (46 %)	5440 (79 %)	268	6513	7360 (63 %)
FF	186	960 (38 %)	10880 (14 %)	172	12198	14720 (15 %)
BRAM	4	0	34 (100 %)	8	46	46 (100%)
DSP	0	8 (75 %)	68 (22 %)	3	79	92 (23 %)
Bitstream size (KB)	-	55	521	-	576	781

FPGA configuration memory. The finer granularity of the hybrid design results in smaller partial bitstreams. Even considering the worst-case scenario where both RP_{FS} and RP_{IFFT} have to be reconfigured, the amount of bitstream data to transfer is still smaller (576 kB) than in the single RP design (781 kB). Thus, apart from requiring less resources, the combination of static multi-mode with DPR-based modules also results in a smaller reconfiguration latency. Another advantage of the hybrid design is the enhanced flexibility enabled by the independent reconfiguration of each datapath module. This allows for the exploration of commonalities among FBMC modes of operation.

5.1.3 Dynamically Reconfigurable Dual-Waveform Modulator: DPR-based *vs.* Static multi-mode designs

The run-time circuit specialization through DPR promises enhanced flexibility, feature wealth, easy system upgradeability, cost-effectiveness and resource/power efficiency (Crockett et al., 2014). Based on this premise, the two previous subsections presented study cases to understand *how* to apply DPR in the design of reconfigurable baseband processors supporting multiple modes of operation. However, to validate the initial premise in the context of this dissertation work, it is crucial to answer the following questions:

1. *Considering FPGA-based systems, how does a DPR-oriented baseband processor compare to a static multi-mode (non DPR-oriented) equivalent design in terms of performance, resource utilization, functional density and power consumption?*

2. *Is the DPR overhead in terms of reconfiguration latency and energy viable/acceptable in the context of modern and future wireless communications?*

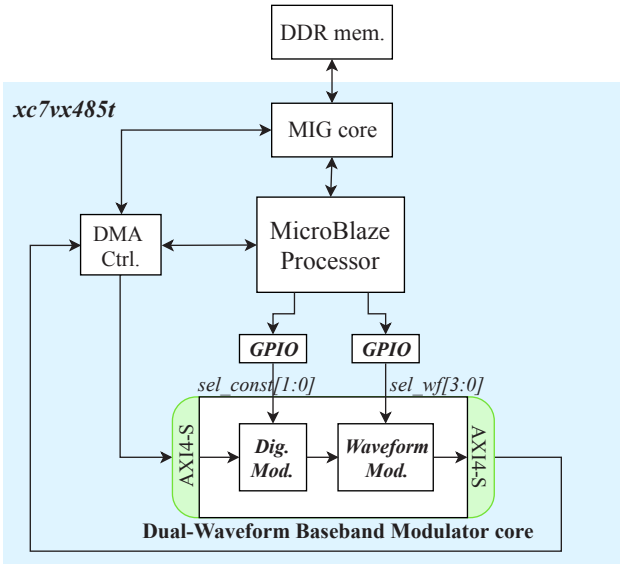
To answer these questions, two functionally equivalent designs for a dynamically reconfigurable dual-waveform baseband modulator were implemented: a static multi-mode design and a DPR-based design. The modulator supports the six OFDM modes of operation defined for LTE downlink transmission (Table 4.1) and one FBMC mode of operation (mode 1 from Table 5.2).

The modulator reconfiguration has two independent planes: the constellation scheme adaptation and the waveform adaptation. The former consists in the change of the QAM modulation scheme used to modulate an input data samples, while the later refers to the modification of the waveform (OFDM or FBMC) or mode of operation (e.g.: OFDM $B_{1.4}$, B_{15} or B_{20}) for baseband processing. Like in Chapter 4, the AXI4-Stream protocol is used on the datapath input/output interfaces and the fixed-point precision in complex arithmetic is 16-bit (Q5.11 format) for real and imaginary parts. In a complete communication system, the implemented baseband processors would be integrated with communication systems like the all-digital transmitters proposed in (Dinis et al., 2016). Here, for evaluation purposes, the dual-waveform modulator core was embedded on a top-level architecture whose basic operation is as follows: 1) read pre-stored input data from DDR memory; 2) perform baseband processing; 3) write output results back in DDR memory. The complete system was implemented in a Xilinx VC707 board, featuring a xc7vx485t FPGA device.

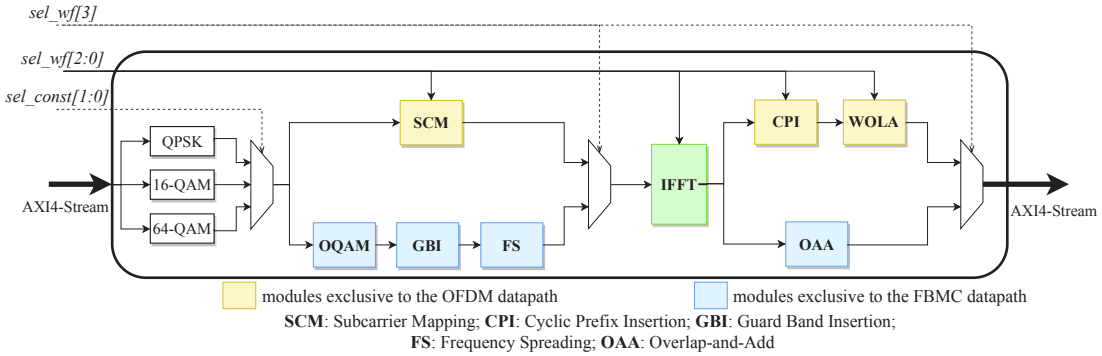
The mode of operation of the dual-waveform modulator is controlled by a *MicroBlaze* soft processor that sets values for control signals and baseband parameters through general purpose input/output (GPIO) interfaces (static multi-mode design) or triggers DPR procedures (DPR-based design). The MicroBlaze also sets data transfers between DDR memory and the dual-waveform modulator. In order to reduce the burden on the MicroBlaze, a 32-bit Direct Memory Access (DMA) controller is used to transfer baseband processing data between the dual-waveform modulator and the DDR memory. This frees the MicroBlaze to execute other system tasks and also improves the data throughput between DDR and FPGA logic. The interface between the FPGA core and the external DDR memory uses the Xilinx MIG (Memory Interface Generator) IP core. The operation and internal structure of the datapaths for OFDM and FBMC modulation were already discussed in section 4.1.1 and 4.2. In this subsection, more emphasis is put on the changes required to design the static multi-mode baseband modulator and the top-level architectures for both static multi-mode and DPR-based designs.

The static multi-mode variant is described first. Figure 5.4 shows the static multi-mode design for the reconfigurable dual-waveform modulator core, as well as the top-level architecture where it is inserted. Constellation scheme adaptation is achieved through a multiplexer structure controlled by the *sel_const* input port. This port drives a 2-bit signal fed by a GPIO interface that is used to select and enable one of the three supported mappers: QPSK (*sel_const = "00"*), 16-QAM (*sel_const = "01"*) or 64-QAM (*sel_const = "10"*). The three mappers are physically present on the system, but only one is used at a time.

Unlike in the digital modulation, the remaining part of the static multi-mode modulator is not the simple instantiation of datapaths for all seven modes of operation considered. To guarantee a fair comparison between the two dual-waveform modulator designs, hardware blocks used in more than one mode of operation were identified and reused. There are two parallel datapaths: one for



(a) Top-level architecture.



(b) Baseband modulation core

Figure 5.4: Static multi-mode dual-waveform modulator design.

Table 5.4: Mode of operation encoding with *sel_wf*

Waveform	mode of operation	sel_wf[3]	sel_wf[2:0]
OFDM	$B_{1.4}$	0	000
	B_3		001
	B_5		010
	B_{10}		011
	B_{15}		100
	B_{20}		101
FBMC	K=4; Nc= 512	1	101

OFDM and another for FBMC baseband modulation. The only common operation across these two datapaths is the IFFT, which is shared between them. The waveform adaptation combines mux/demux structures with parameter value setting and is controlled by a 4-bit signal from the *sel_wf* input port (Table 5.4). The most significant bit of *sel_wf* selects and activates one of the two datapaths (OFDM or FBMC). As for the *sel_const* signal, the value of *sel_wf* is received through a GPIO interface controlled by the MicroBlaze.

The OFDM modules for subcarrier mapping, CP insertion and WOLA were dimensioned for the most resource-demanding scenario and their operation parameters are defined by the three least significant bits of *sel_wf*. For FBMC, the datapath modules, excluding the IFFT, were designed for the only mode of operation considered ($K = 4$, $N_c = 512$, IFFT size $= 2048$). To support all required IFFT sizes, the IFFT also combines mux/demux structures with parameter value selection through *sel_wf[2:0]*. Hardware structures common to different IFFT sizes were identified and reused. Still, modules for data reordering and Radix-2 computation were designed for the most resource-demanding scenario. Figure 5.5 illustrates the static multi-mode IFFT core and lists the modules used for each IFFT size. Despite the concerns about hardware reuse during the design of the static multi-mode core, only one of the two datapaths is used at a time and several modules had to be designed for the most demanding scenario.

In contrast to the static multi-mode OFDM modulator, the DPR-based dual-waveform modulator optimizes its baseband processing datapath on-the-fly for each mode of operation. As constellation scheme adaptation is independent from waveform adaptation, a straightforward datapath partition

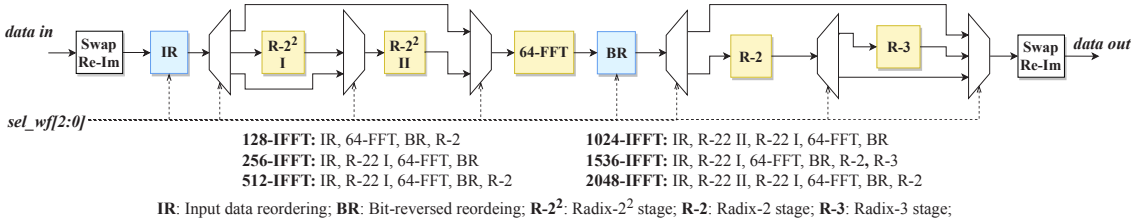


Figure 5.5: Static multi-mode IFFT core structure.

strategy would consider one RP for constellation scheme adaptation and another RP for waveform processing. However, the QAM mapper has very low resource complexity and would not benefit significantly from DPR application. For this reason, it was decided to reuse the QAM mapper architecture from the static multi-mode design.

From subsections 5.1.1 and 5.1.2, a single RP approach is a better choice for a dynamically reconfigurable OFDM modulator, while a hybrid design is more advantageous for a dynamically reconfigurable FS-FBMC modulator. Although only two waveforms are considered in this subsection, the DPR-based architecture is meant to be upgradeable with new waveforms or modes of operation. Thus, considering a finer granularity level for reconfiguration would bias the architecture and turn it less general. For this reason, the reconfiguration of the DPR-based modulator design targets the whole processing datapath, rather than each individual module, and a single RP approach is adopted. As a result, the waveform adaptation requires the generation of seven partial bitstreams: six for OFDM modes of operation and one for the FBMC mode of operation. During DPR, signals driven by the RP into static logic can assume unpredictable values which, in turn, can corrupt normal system functioning. Likewise, signals driven by static logic into an RP that is being reconfigured may corrupt the new module loaded into the RP. To avoid these problems, registers are placed on the static part to decouple the signals generated from and driven into the RP.

For the DPR-based design, the top-level architecture (Figure 5.6) is slightly different from that of the static multi-mode design, as support for DPR is required. Before the initial FPGA configuration, the board flash memory is loaded with a file system that includes all partial bitstreams and input data files. Then, after the initial FPGA configuration, the MicroBlaze copies all partial bitstreams from the flash memory to the DDR. Once this is completed, the system is initialized and can start normal operation. During system operation, the MicroBlaze is also responsible for triggering DPR procedures. In our system, the access to configuration memory functions is provided by the Xilinx Internal Configuration Access Port (ICAP) primitive. The partial bitstream transfers from the DDR to the ICAP are set up by the MicroBlaze, but controlled by a dedicated 32-bit DMA controller. The use of this dedicated DMA controller is motivated by the need to accelerate the provision of partial bitstreams to the ICAP and, consequently, the reduction of DPR latency overhead. Recalling the general DPR architectural model from Figure 2.12, the DDR memory and the DMA controller can be viewed as the high-speed external memory and the on-chip memory buffer, respectively.

The ICAP primitive has a 32-bit input port and also a 32-bit configuration data output that can be used to report and monitor the configuration status. Based on the ICAP output port values (Xil, 2015b), two flag signals are generated: *pr_dec* and *pr_error*. The former indicates whether or not the reconfiguration of an RP is completed and if so, it resets the decoupling registers placed between the static logic and the RP; the later informs the MicroBlaze if a configuration error has occurred during a DPR procedure.

The two dual-waveform modulator designs are now evaluated and compared. Following the design guidelines presented in Chapter 4, both baseband modulator designs are able to produce one sample per clock cycle after an initialization processing latency. For each mode of operation the active datapath used for baseband processing is equivalent and, consequently, the initial processing

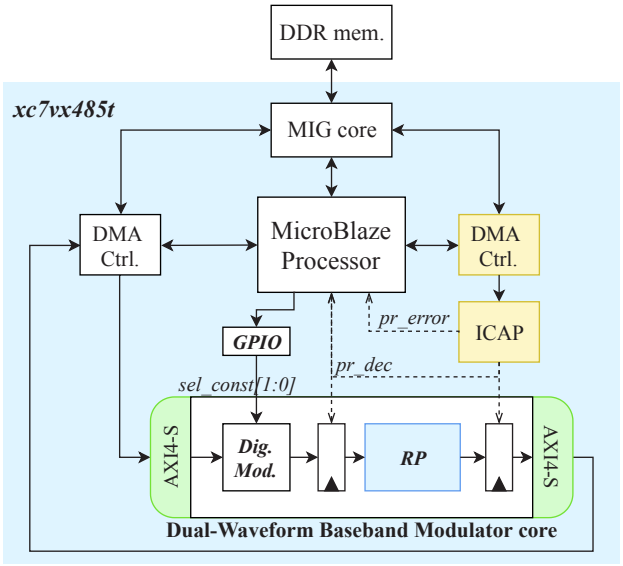


Figure 5.6: Top-level architecture for the DPR-based design

latency is the same for both static multi-mode and DPR-based design. For OFDM, the worst-case latency is 9278 clock cycles (B_{20} mode) and for FBMC it is 7743 clock cycles. The processing throughput was also measured and, for a clock frequency of 100 MHz, both static multi-mode and the DPR-based designs achieve a data throughput of at least 92 MSample/s. These throughput values were calculated by the bare-metal application running on the MicroBlaze. It measures the time duration of the DMA operation, from sending input data to the baseband processing to writing the results back in the DDR. The throughput is then calculated as the ratio between the number of the output baseband samples produced and the time duration of the DMA operation.

It was observed that, for the same clock frequency, the static multi-mode and the DPR-based modulator cores show the same processing throughput. Another possible analysis is the evaluation of the estimated maximum clock frequency supported by each modulator design. The timing reports issued by Xilinx Vivado 2015.2 do not explicitly state the estimated maximum clock frequency (f_{max}) supported by a design. However, it is still possible to estimate it using the following expression:

$$f_{max} = \frac{1}{T_{clk} - WNS}, \tag{5.1}$$

where T_{clk} is the period of the clock signal used for implementation purposes (10 ns in this case) and *WNS* is the *worst negative slack* associated with the design critical path. Table 5.5 presents the f_{max} values for the static multi-mode baseband modulator and for each mode of operation of the DPR-based core. The f_{max} for the static multi-mode case is between 7.3 and 15.9 MHz lower than in the DPR-based modes of operation. This indicates that the datapath specialization may contribute to optimized floorplaning and to circuits that can operate at higher clock frequencies.

The throughput values measured and enabled by the maximum clock frequencies estimated for our designs are compatible with the data rates imposed by modern communication requirements. As there are no FBMC-based proposed standards, OFDM-based numerologies are used in this

Table 5.5: Estimated maximum clock frequency for the static multi-mode (STA) and DPR-based (DPR) dual-waveform modulator cores. The WNS values refer to the critical path within the modulator core. Post Place-and-Route results using Vivado 2015.2; $f_{clk} = 100\,\text{MHz}$; Device: xc7vx485t.

		WNS (ns)	f_{max} (MHz)
STA		1.717	120.729
DPR	$B_{1.4}$	2.494	133.227
	B_3	2.679	136.593
	B_5	2.429	132.083
	B_{10}	2.643	135.925
	B_{15}	2.188	128.008
	B_{20}	2.287	129.651
	FBMC	2.598	135.099

analysis. The sampling rate requirements imposed by LTE depend on the mode of operation and its highest value is 30.72 MSample/s (for B_{20}). Moreover, without further datapath optimization, the presented designs can also support the one of the OFDM numerologies defined in (TS, 2018) for 5G, whose sampling frequency is 61.44 MHz.

Next, the resource utilization is analyzed at different levels: first, the resources used for constellation and waveform modulator cores are quantified and discussed; second, the resource utilization for the top-level architecture (without constellation and waveform modulator cores) in both static multi-mode and DPR-based designs is analyzed. This two-step analysis provides a better perspective about resource usage in each design. The resource utilization results from sections 4.1.3 and 4.2.2 considered the independent place and route of OFDM and FBMC baseband modulators. In this subsection, the baseband modulators are part of a complete embedded system and their floorplaning and routing is also constrained by the rest of the embedded system design. Therefore, resource utilization is again quantified in the context of the two dual-waveform modulator designs implemented.

The mux-based QAM mapper uses 43 LUTs and 40 FFs in both static multi-mode and DPR-based designs. This represents a very low resource usage and supports the decision to keep a static multi-mode constellation mapper in the DPR-based design. Table 5.6 quantifies resource utilization for all variants of the waveform modulation core. Each of the seven configurations of the DPR-based design uses less resources than the static multi-mode waveform modulator core. The resource savings are mainly due to the circuit specialization achieved through DPR and become more evident for lower channel bandwidth OFDM modes. In the DPR-based design, *OFDM* B_{15} mode of operation is the most demanding in terms of LUTs (4070), FFs (3128) and DSPs (23), whereas the most demanding configuration regarding BRAM tiles (21) is the *FBMC* mode. For each resource type, a comparison to the static multi-mode design shows that the DPR-based waveform modulator achieves a resource utilization reduction of at least 42% of LUTs, 28% of FFs, 31% of BRAM tiles and 28% of DSP blocks.

Table 5.6: Post Place-and-Route resource utilization for the waveform modulator core. $f_{clk} =$ 100 MHz; Device: xc7vx485t.

Resource	Available	Static multi-mode	DPR-based (implemented in RP)						
			$B_{1.4}$	B_3	B_5	B_{10}	B_{15}	B_{20}	FBMC
slice	75900	2793	850	937	971	1099	1297	1173	1148
LUT	303600	6961	2279	2604	2795	3368	4070	3780	3521
FF	607200	4367	1929	1966	2133	2196	3128	2612	2332
BRAM	1030	30.5	3	4	7	10.5	17	18	21
DSP	2800	32	11	11	14	14	23	17	21

These resources saving of the DPR-based design must be weighted against the overhead on the top-level architecture related to the extra modules to perform DPR. The main differences between the top-level architectures for static multi-mode and DPR-based designs reside in the inclusion of the ICAP primitive and a dedicated DMA controller for partial bitstream data transfers. The additional DMA controller for DDR-ICAP transactions makes the IP core responsible for the interconnection between the MIG core (master device) and slave devices (MicroBlaze, DMA controllers) more complex and resource demanding. A comparison of the resources used by the top-level architecture in both designs is presented in Table 5.7. The top-level architecture for the DPR-based design requires more 1605 LUTs, 1466 FFs and 9.5 BRAM tiles than in the static multi-mode desing. However, combining the values from Tables 5.6 and 5.7, one observes that, except for FFs in the B_{15} mode, this overhead is compensated or surpassed by the savings obtained from circuit specialization in all DPR-based design configurations.

Another overhead of the DPR-based design is the amount of resources reserved for the RP, as it has to be dimensioned for its most-resource demanding configuration. The resources reserved by the waveform modulation RP are shown in Table 5.8. Regarding the RP resource utilization, the maximum slice and BRAM tiles utilization is about 93% and 70%, respectively, whereas for the DSP blocks case it drops to about 38%. Combining the figures for slices, BRAM tiles and DSP blocks, the total amount of resources *reserved* for the RP in the DPR-based design is about half the amount of resources *used* by the static multi-mode waveform modulation core. This clearly attests the improvement in resource efficiency enabled by DPR in our application.

Table 5.7: Post Place-and-Route resource utilization for the Top-level System Architecture (without the waveform modulator core). The figures in parenthesis correspond to the percentage of available FPGA resources in use.

Resource	Static multi-mode	DPR-based
slice	9439 (12.44%)	10402 (13.70%)
LUT	27703 (9.12%)	29308 (9.65%)
FF	20837 (3.43%)	22303 (3.67%)
BRAM	95.5 (9.27%)	105 (10.19%)
DSP	1 (0.04%)	1 (0.04%)

Table 5.8: Resource reservation for the Reconfigurable Partition (R - Reserved; MU - Maximum Utilization) and its comparison with the number of resources used by the Static Multi-mode waveform modulator core.

Resource	RP		Static Multi-mode
	R	MU	Used
slice	1400	1297	2793
BRAM	30	21	30.5
DSP	60	23	32
Total	1490	1341	2855.5

In the static multi-mode implementation, the system reconfiguration consists in defining new values for *sel_const* and *sel_wf*, which has no significant impact on the normal system operation. In contrast, the reconfiguration of the waveform modulation in the DPR-based implementation has a latency associated. During the DPR process, the RP to be reconfigured is not available and, if too big, the reconfiguration latency can have a negative impact on system performance. Thus, it is important to reduce the DPR latency and evaluate its impact on the considered application domain. In order to reduce the partial bitstreams size and DPR latency, the bitstream compression feature offered by Vivado (Xil, 2015b) was used to generate the partial bitstreams. It is worth noting that the size of compressed bitstreams for the same RP can vary. The DPR latency is the time elapsed during the DMA operation that fetches a partial bitstream from the DDR memory sends it to the ICAP. It was measured by the MicroBlaze bare metal application. The worst-case DPR latency for waveform adaptation is 1.051 ms and corresponds to reconfiguring the RP to the *OFDM* B_{20} mode. It involves the transfer of 383 kB to the ICAP. The ICAP primitive is clocked at 100 MHz, which means that the upper limit of the reconfiguration throughput is 400 MB/s. The lowest reconfiguration throughput registered for waveform adaptation was 369 MB/s, which is 92.25% of the ICAP limit. Compared to the 5G requirements defined in (ITU-R, 2017), the worst-case DPR latency measured is around 10% of the control plane latency requirement (10 ms), leaving a considerable margin for other control plane tasks to be performed in the radio system. This observation shows the viability of exploiting DPR in the design of baseband processing hardware infrastructures for wireless systems.

The results for resource utilization and DPR latency allow us to perform an analysis based on the *functional density* (D) metric proposed by Wirthlin and Hutchings to evaluate "*the advantages of RTR against its associated reconfiguration costs*" (Wirthlin and Hutchings, 1998). Functional density is defined as:

$$D = \frac{1}{A(T_e + T_r)}, \tag{5.2}$$

where T_e is execution/operation time under a certain configuration, T_r is the maximum reconfiguration latency and A is the amount of hardware resources to implement the required functionality. For

the static multi-mode design, the functional density is:

$$D_s = \frac{1}{A_s T_e} \tag{5.3}$$

as there is no reconfiguration latency ($T_r = 0$). The functional density for the DPR-based design is:

$$D_r = \frac{1}{A_r(T_e + T_r)}. \tag{5.4}$$

In these expressions, A_s and A_r are measured as an aggregate amount of slices, BRAMs and DSPs. A_s is the amount of resources *used* in the static multi-mode design and A_r is the amount of RP *reserved* resources in DPR-based design. For T_e, we consider it to be the same in both expressions because both static multi-mode and DPR-based designs showed the same processing throughput performance, considering the same clock frequency. Although it is hard to know for how long a device will operate in a certain mode of operation, it is reasonable to admit that, in our application scenario, typical values of T_e are in the range from minutes to tens of seconds. Wirthlin and Hutchings (Wirthlin and Hutchings, 1998) consider that a DPR-based design has a higher functional density than its static multi-mode counterpart when:

$$\frac{D_{rmax}}{D_s} - 1 \geq \frac{T_r}{T_e} \Leftrightarrow \frac{A_s}{A_r} - 1 \geq \frac{T_r}{T_e}, \tag{5.5}$$

where D_{rmax} is the upper limit for the DPR-based design functional density (when $T_r = 0$): $D_{rmax} = 1/A_r T_e$. For the waveform modulator core, $A_s = 2855.5$, $A_r = 1490$ (Table 5.8) and $T_r = 1.051$ ms and we have:

$$0.916 \gtrsim \frac{T_r}{T_e} \Leftrightarrow T_e \gtrsim 1.147\,\text{ms}. \tag{5.6}$$

Thus, if T_r is less than about 92% of T_e, the DPR-based waveform modulator shows a superior functional density than its static multi-mode counterpart. In other words, it states that when T_e is bigger than 1.147 ms, $D_r \geq D_s$. For the typical values of T_e previously mentioned in this section, condition 5.6 is satisfied and the DPR-based design is superior than the static multi-mode design in terms of functional density.

The power analysis is divided in two parts: first, power consumption estimates for the static multi-mode and DPR-based design are presented and compared; second, the DPR of energy overhead is quantified through real-time power consumption measurements. Power estimates for both static multi-mode and DPR-based modulator core designs were obtained using the power analysis capabilities of Vivado 2015.2. To obtain high confidence estimates, node switching activity values were derived from post Place-and-Route simulation of fully routed netlists. The static power consumption shows no significant variations across different modes of operation and it is very similar in both designs: 721 mW in the static multi-mode design; 720 mW in the DPR-based design. As the FPGA device size is considerably larger than the area used by the modulator cores, the static power consumption is dominant and represents between 68% to 78% of the total power consumption.

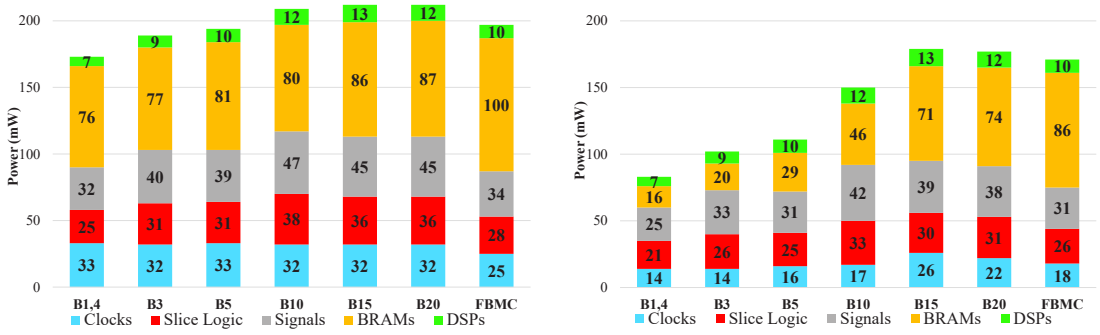
Table 5.9: Dynamic power consumption estimates for both dual-waveform modulator core designs. $f_{clk} = 100\,\text{MHz}$; Device: xc7vx485t

	Static Multi-mode	DPR-based	Power Reduction
OFDM $B_{1.4}$	173 mW	83 mW	90 mW (52%)
OFDM B_3	189 mW	102 mW	87 mW (46%)
OFDM B_5	194 mW	111 mW	83 mW (43%)
OFDM B_{10}	209 mW	150 mW	59 mW (28%)
OFDM B_{15}	212 mW	179 mW	33 mW (16%)
OFDM B_{20}	212 mW	177 mW	35 mW (17%)
FBMC	197 mW	171 mW	26 mW (13%)

The dynamic power consumption shows considerable variations across modulator core designs and modes of operation. In general, the dynamic power consumption is higher for modes of operation with higher resource utilization (Table 5.9). This observation is more evident for the DPR-based implementation. For all modes of operation, the DPR-based modulator core presents a reduced dynamic power consumption compared with its static multi-mode counterpart. These power savings vary from 26 mW (13% reduction in the *FBMC* mode) to 90 mW (52% reduction in the *OFDM* $B_{1.4}$ mode). Figures 5.7a and 5.7b present the consumption per on-chip component type for the static multi-mode and DPR-based modulator cores, respectively. The power savings in the DPR-based design are mainly associated with clock signal propagation and BRAM primitives utilization. The use of clock gating techniques in the static multi-mode core could reduce the clock-related dynamic power. However, due to the fixed nature of the FPGA clock trees, clock gating can become challenging and potentially affect design timing closure. Regarding the BRAM dynamic power, it is difficult to reduce it because the static multi-mode modulator core has to be dimensioned for the most resource-demanding scenario. In turn, the run-time circuit specialization afforded by the DPR-based implementation reduces the amount of node activity within the baseband modulator core, leading to significant dynamic power savings.

For the DPR-based implementation, the energy overhead introduced by DPR should be measured and its impact on the system should be evaluated. Liu et al. (Liu et al., 2009) state that the DPR energy overhead can be mitigated by maximizing of the reconfiguration speed. However, the DPR energy overhead is not only influenced by the reconfiguration speed. Bonamy et al. (Bonamy et al., 2014) observed that the differences between the previous and the next configuration also affect the power consumption behavior during reconfiguration. It was observed that the Hamming distance between the bitstreams for the previous and next configurations has a strong correlation with the over-consumption during reconfiguration.

The simulation and power estimation for DPR are very challenging. Moreover, DPR is a system-level feature and its energy impact should be evaluated regarding the whole system design. In the VC707 board, the power rail that feeds the FPGA core provides a voltage of 1 V (V_{FPGA}). The current fed by this power rail can be measured by a circuit setup available on the board that



(a) Static multi-mode

(b) DPR-based

Figure 5.7: Dynamic power consumption estimates per on-chip component for the dual-waveform baseband modulator cores.

consists of a current-sensing shunt resistor (5 mΩ ±1%, 3 W) and a Texas Instruments INA333 instrumentation amplifier. The current driven into the FPGA core is the same that flows through the shunt resistor and the instrumentation amplifier configuration has a gain $G = 24.7$. Thus, the current through the shunt resistor can be obtained from the INA333 output voltage - V_o - by the following relation:

$$I_R = \frac{V_o}{G \times 5\,\text{m}\Omega} \qquad (A). \tag{5.7}$$

In turn, the FPGA core power consumption is given by:

$$P = V_{FPGA} \times I_R \qquad (W). \tag{5.8}$$

The INA333 output voltage was measured with a Keysight Infiniium DSO90254A 2.5 GHz digital oscilloscope. Data was acquired from the oscilloscope at a sampling rate of 1.25 MSample/s and, during all experiments, room temperature and FPGA device temperature were kept around 23 °C and 33 °C, respectively. From the INA333 output voltage, the FPGA core real-time power consumption was calculated according to 5.8.

Both DPR latency and differences between partial bitstreams were considered, when defining a scenario for the worst-case DPR energy overhead. As pointed out previously, the worst-case DPR latency is 1.051 ms, when reconfiguring the waveform modulation RP to the *OFDM* B_{20} mode. Using non-compressed partial bitstreams, Hamming distances between waveform configurations were computed. The largest Hamming distance observed is between the *OFDM* B_{20} and B_{15} modes. Consequently, the scenario considered to measure the DPR energy overhead was the reconfiguration between *OFDM* B_{20} and *OFDM* B_{15} modes. For measurement purposes, the system operation loop was: remain 4.5 ms in the *OFDM* B_{20} mode (without baseband processing); reconfigure to the *OFDM* B_{15} mode; remain 2.5 ms in the *OFDM* B_{15} mode (without baseband processing); reconfigure to the *OFDM* B_{20} mode. The power consumption curve from Figure 5.8 shows over-

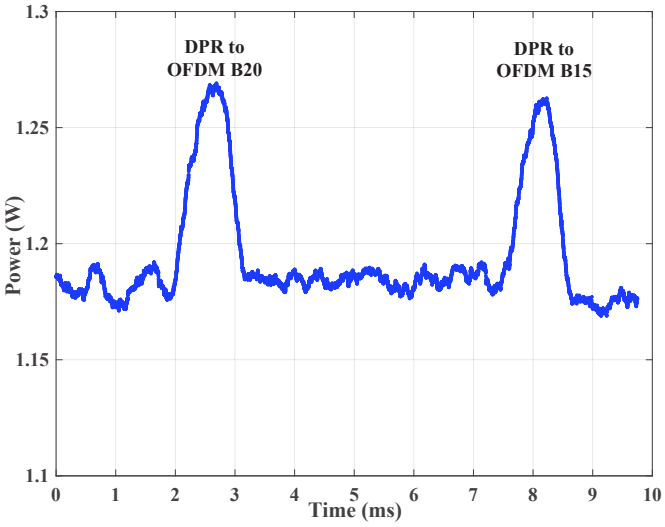


Figure 5.8: Power consumption behavior during DPR (real-time measurement).

consumption periods of time that corresponds to DPR procedures. The duration of these peaks matches the reconfiguration latency for *OFDM* B_{20} and *OFDM* B_{15} (approximately 1 ms).

In this application domain, the reconfiguration is a rather sporadic event with a very short duration compared to the periods of time where the system remains in a specific mode of operation. For this reason, the DPR impact is quantified in terms of energy consumption. Table 5.10 presents values for DPR latency and energy consumption overhead, as well as the partial bitstream sizes for the reconfiguration scenarios considered. The DPR energy overhead is slightly higher when reconfiguring to *OFDM* B_{20}, probably due to the higher latency associated. Overall, the DPR energy overhead is below 1.5 mJ for both configuration transitions. As mentioned in (Moy and Palicot, 2015), FPGAs are particularly suited to implement multi-standard, flexible and reconfigurable radio devices for base stations, where the power consumption constraints are less stringent compared to user terminal equipment. Occurring sporadically and with an overhead of few mJ, the DPR energy consumption overhead is acceptable in the context of commercial base stations.

Based on the results for the dual-waveform modulator, the questions formulated earlier in this subsection can now be answered:

1. *Compared to a static multi-mode equivalent design, the DPR-based design proved to be superior: it can operate at higher clock frequencies; it reserves less resource than those used*

Table 5.10: DPR latency and energy consumption overhead

Reconfigure to	OFDM B_{20}	OFDM B_{15}
Partial bitstreams	383 kB	363 kB
DPR latency	1.051 ms	0.993 ms
DPR energy	1.31 mJ	1.23 mJ

by its static multi-mode counterpart; it is characterized by a higher functional density in the considered application domain; and it allows for dynamic power consumption savings.

2. *With reconfiguration latency and energy overhead in the order of few ms and mJ, respectively, DPR is a viable FPGA-oriented technique to design flexible baseband processors for modern and future wireless devices, especially for commercial base stations.*

The considered dual-waveform design only supports one of seven modes of operation. But even with a relatively limited range of modes of operation, the advantages of DPR become clear. The upgradeability is hard to quantify, but it is another aspect in favour of the DPR-based design. For instance, let us assume that the FBMC mode 3 ($K = 2$, $N_c = 1024$) must be supported by the dual-waveform modulator. In the static multi-mode, that would require the redesign of the baseband modulator core. In contrast, for the DPR-based design, as the RP provides enough resources to implement the new mode, it is only necessary to design the specialized circuit for FBMC mode 3. The extension of the DPR-based design functionality is limited by the reserved RP resources. Increasing the amount of supported modes of operation immediately increases the benefit of the DPR-based design in terms of resource efficiency and functional density. So, the advantages of the DPR-based design tend to become more evident when the heterogeneity of numerologies to be supported increases, which is a very likely scenario in future communications.

5.1.4 Discussion

This section explored the specialization of computation at run-time through the application of DPR. The first two dynamically reconfigurable systems uncovered the design-time challenges of applying DPR to baseband processing datapaths. The definition and sizing of reconfigurable partitions requires a careful analysis of the resource utilization and baseband parameter variation across different modes of operation. The dynamically reconfigurable modulators for OFDM (subsection 5.1.1) and FS-FBMC (subsection 5.1.2) show that the best datapath partitioning strategy may not be the same for every case. Consequently, the application of DPR should be individually studied for each target scenario.

Subsection 5.1.3 compares two design strategies to implement a reconfigurable baseband modulator supporting OFDM and FBMC numerologies. The first design relies on parameter value setting and mux-demux structures to adapt the baseband mode of operation - *static multi-mode design*. The second design employs DPR to adapt the modulator circuitry according to the immediate communication demands - *DPR-based design*. Results show that the DPR-based design can operate at clock frequencies 7.3 MHz (6%) to 15.9 MHz (13%) higher than the equivalent static multi-mode design, while reserving around half the aggregate amount of slices, BRAMs and DSPs. The increased resource efficiency confers an improved functional density to the DPR-based design. The circuit specialization on the DPR-based design also translates into dynamic power savings between 13% (26 mW) to 52% (90 mW) compared to the static multi-mode design. On top of that, the nature of the DPR design allows for an easy extension of the modulator functionality for other numerologies, making the architecture *forward-compatible*.

Table 5.11: Class of OFDM numerologies for 5G NR data transmission(3GPP Release 15) (TS, 2018). Δf: subcarrier spacing ($2^{\mu} \times 15\,\text{kHz}$); A: number of active subcarriers per OFDM symbol; N: IFFT size; L_{CP}: cyclic prefix length; $f_{sampling}$: sampling rate ($N \times \Delta f$).

μ	0	1	2	3
Δf (kHz)	15	30	60	120
A	3300			
N	4096			
L_{CP}	288			
$f_{sampling}$ (MHz)	61.44	122.88	245.76	491.52

The current section is one of the cornerstones of this dissertation. It validates DPR as a convenient design technique to achieve flexible, resource/power efficient baseband processors and shows that the DPR overhead in terms of reconfiguration latency (few ms) and energy (few mW) is tolerable in modern and future communications.

5.2 The Specialization of Performance at Run-Time

The 3GPP Release 15 for 5G New Radio (NR) specification (TS, 2018) has recently proposed a family of OFDM numerologies to enable the communication for different use case scenarios and using a wide range of frequencies. This includes previously unused spectrum bands above 6 GHz (e.g.: millimeter waves). This family of numerologies shares some aspects with LTE: the normal CP represents a symbol extension of about 7% and a physical resource block (PRB) is formed by 12 subcarriers. However, while LTE fixes the subcarrier spacing (Δf) to 15 kHz, 5G NR considers subcarrier spacing values that are power of two multiples of 15 kHz: $\Delta f = 2^{\mu} \times 15\,\text{kHz}$, where μ is an integer that specifies the mode of operation. In 5G NR, a maximum of 3300 active subcarriers are transmitted per OFDM symbol, whose baseband modulation can be efficiently performed with a 4096-IFFT (the remaining $4096 - 3300$ subcarriers are used as guard bands). Table 5.11 presents 5G NR numerologies for data transmission, considering $\mu = \{0, 1, 2, 3\}$. In this case, the switching between modes of operation only requires the clock frequency adaptation to cope with the desired sampling frequency, as baseband parameters like IFFT size and CP length remain the same. The challenge here is to have an OFDM baseband modulator architecture, whose performance can be scaled to support the range of requirements from Table 5.11 in a power-efficient way. This demands specialization for *performance*, rather than *functionality*, at run-time.

The OFDM baseband modulator from section 4.1.1 follows a pipelined architecture and produces a steady-state throughput of one sample per clock cycle. This architecture may be suitable for 5G NR modes with $\mu = \{0, 1\}$. However, to support $\mu = 3$, the OFDM modulator would have to run at a clock frequency around 500 MHz. Despite the claims from (Pecot, 2014), the design of FPGA-based digital signal processors to achieve such high clock frequencies is very challenging. Moreover, not all modes of operation demand such high performance, leading to

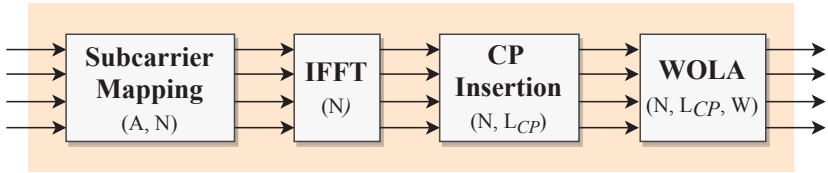


Figure 5.9: General datapath structure for the parallel-pipelined OFDM modulator.

system over-optimization and high power consumption. According to Luo and Zhang, to achieve high-data throughput with reasonable clock frequencies, FPGA-oriented "*baseband processing architectures must be designed to operate on arrays of samples in each clock cycle*" (Luo and Zhang, 2016). Then, the multiple output samples produced are fed to a set of low-frequency interleaved DACs.

Two cases can be identified here, depending on how the array samples relate to each other. If the samples within the array are independent from each other, multiple baseband processors can be used to process each sample in parallel. For example, this occurs in Multiple Input Multiple Output (MIMO) communications, where multiple independent data streams are processed in parallel by an array of baseband processors. However, if the baseband samples to process in a clock cycle belong to the same data sequence, the baseband processor architecture must be able to process them in a combined way. Garrido et al. (Garrido et al., 2013) identified this type of processing as *parallel-pipelined* and studied its application to FFT/IFFT cores. This section contributes an FPGA-oriented architecture for a parallel-pipelined OFDM baseband modulator supporting 5G NR numerologies and whose performance can be adapted at run-time through *dynamic frequency scaling* (DFS). The power consumption benefits of DFS, as well as the clock adaptation latency are measured and discussed. Unlike (Garrido et al., 2013; Luo and Zhang, 2016), the parallel-pipelined architecture is applied not only to the IFFT core, but also to subcarrier mapping, CP insertion and WOLA. To our best knowledge, the extension of the parallel-pipelined concept to the complete OFDM baseband modulator datapath is a novelty.

A parallel-pipelined OFDM modulator receives A QAM modulated samples (active subcarriers) to produce an OFDM symbol via s parallel input streams. Taking into account the dependencies between them, these s streams are concurrently processed to produce time-domain OFDM symbols. Each resulting OFDM symbol is delivered in s parallel streams. The design presented here considers four pipelined streams ($s = 4$), but its extension for other of powers-of-two values of s is straightforward. Apart from the baseband parameters from Table 5.11, is is assumed that the number of time-domain samples for WOLA (parameter W) is 32 and, as in previously presented baseband datapaths, 16-bit fixed-point precision with Q5.11 format for real and imaginary components is used. The parallelization of each datapath module is described next and, for each module, the input/output stream sizes refer to the amount of samples needed to produce one OFDM symbol. For easy referencing, the general datapath structure for OFDM modulation is again reproduced, now considering 4 input/output parallel streams (Figure 5.9).

As in section 4.1.1, subcarrier mapping maps the A input active subcarriers to the central bins

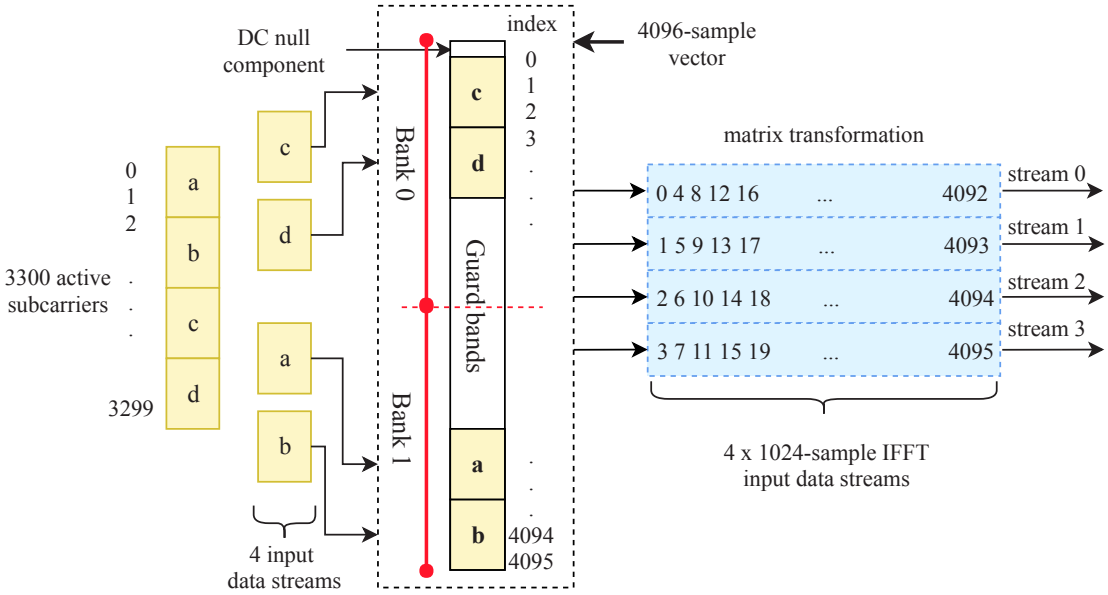


Figure 5.10: Basic operation of the parallel-pipelined *subcarrier mapping* module

of an N-element array, zeroes the DC null subcarrier and performs an IFFT shift operation. In the parallel-pipelined architecture, the A (3300) active subcarriers are received as four contiguous blocks of $A/4$ (825) samples. Figure 5.10 illustrates the operation of subcarrier mapping in the parallel-pipelined architecture. For each OFDM symbol, a 4096-element array is formed and delivered to the IFFT core as four streams of 1024 ($4 \times N/4$) elements each. The index mapping used to create the four subcarrier mapping output streams is derived from (Meyer-Baese, 2007) and aims at simplifying the IFFT and subsequent time-domain operations (CP insertion and WOLA).

Subcarrier mapping in section 4.1.1 was performed using a double-buffer memory structure and two memory operations per clock cycle: one to write a new sample in one half of the double buffer; and one to read a sample from the the other half of the double buffer. But here, the subcarrier mapping module has to performs 4 write and 4 read operations on the double-buffer per clock cycle. From Figure 5.10, one realizes that two write operations always occur in the first half of the 4096-element array, while the other two write operations occur in the second half of the array. However, the four read operations can occur in the same half of the 4096-element array. The memory structure to implement the 4096-element array combines *banking* and *replication*, two of the conventional multi-port memory techniques presented in (LaForest and Steffan, 2010). The 4096-element array is divided into two 2048-elements *banks*. Each bank is implemented as a true dual-port BRAM with two write ports, following the synthesis guidelines from Xilinx (Xil, 2015a). To allow four read operations from the same bank, each true dual-port BRAM is replicated. So, the 4096-element array is implemented as BRAM-based quad-port memory made of 2×2 true dual-port BRAMs of 2048-elements. As in section 4.1.1, to simultaneous read and write of consecutive A-element arrays without data conflicts a double-buffer structure is used and, consequently, the 4096-element quad-port memory is duplicated. Additionally, a control unit manages the read/write addressing of the double-buffer structure.

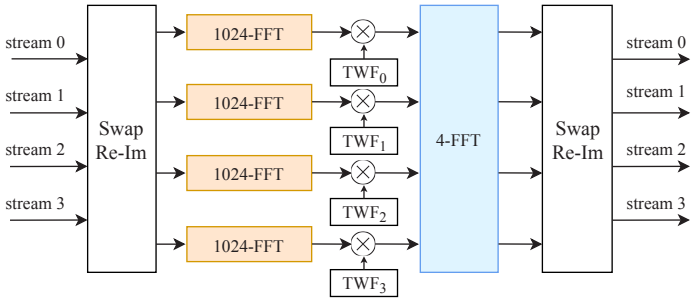


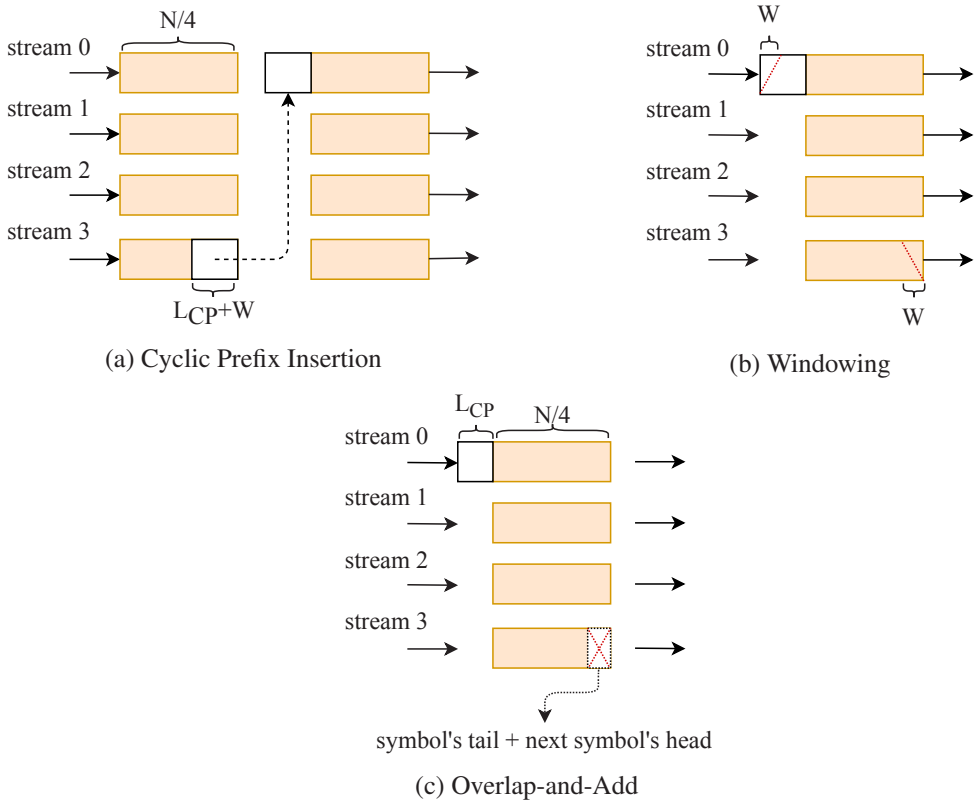
Figure 5.11: Internal structure of the parallel-pipelined IFFT module

The four 1024-sample output streams of the subcarrier mapping module are the input streams of the IFFT. Recalling the Cooley and Tukey FFT algorithm discussed in section 2.2 and assuming a $4096 = 1024 \times 4$ factorization, a 4096-FFT core can be decomposed into four parallel 1024-FFT cores and one 4-FFT core (Figure 5.11). Each IFFT input stream directly feeds a 1024-FFT core. The intermediate results from the four 1024-FFT cores are *rotated* by twiddle factor multiplications and fed to a 4-input/output Radix-4 butterfly that implements the 4-FFT core. The architecture of the 1024-FFT follows the pipelined FFT architecture presented in section 3.1, while the 4-FFT core results from the direct implementation of Radix-4 butterfly expressions from (Lin and Chung, 2013).

Because of the input/output index mapping used, the IFFT output streams appear in natural order and correspond to contiguous blocks of the time-domain OFDM symbol. This greatly simplifies the parallel-pipelined version of the CP insertion and WOLA modules, as their operations only involve the beginning and the end of an OFDM symbol. In other words, only IFFT output streams 0 and 3 need to be processed through CP insertion and WOLA. Streams 1 and 2 remain unchanged and just need to be synchronized with streams 0 and 3, using storage elements.

Details on the CP insertion and WOLA operation were previously discussed in sections 2.1.1 and 4.1.1. Their adaptation to the parallel-pipelined architecture is illustrated in Figure 5.12. The CP insertion module copies the last $L_{CP}+W$ (288+32) samples of stream 3 and prepends them the beginning of stream 0. This requires the temporary storage of streams 0 and 3, in order to extend the stream 0 by $L_{CP}+W$ samples. For synchronization purposes, streams 1 and 2 are also temporarily stored. BRAM-based memory structures and finite state machines manage the storage and forwarding of the parallel streams. At the output of the CP insertion module, stream 0 is extended to $N/4+L_{CP}+W$ (1344) samples, while the other streams keep the same size (1024).

In the first step of the WOLA operation, the the first W (32) samples of the input stream 0 and the last W samples of the input stream 3 are multiplied by non-unitary raised-cosine window coefficients that are pre-stored in a ROM memory (32 elements of 16-bit). Like in the pipelined version of the OFDM modulator (section 4.1.1), two real multipliers (DSP-based) are required to separately window the real and imaginary components, as well as mux-demux structures and a control FSM. The FSM used to control of the overlap-and-add operation in the parallel-pipelined architecture follows a very similar approach to the one designed for the pipelined architecture (section 4.1.1). Figure 5.13 depicts the architecture of the WOLA module for the parallel-pipelined



(a) Cyclic Prefix Insertion

(b) Windowing

(c) Overlap-and-Add

Figure 5.12: Basic functioning of the time-domain operations: *CP insertion* and *WOLA* (Windowing and Overlap-and-Add).

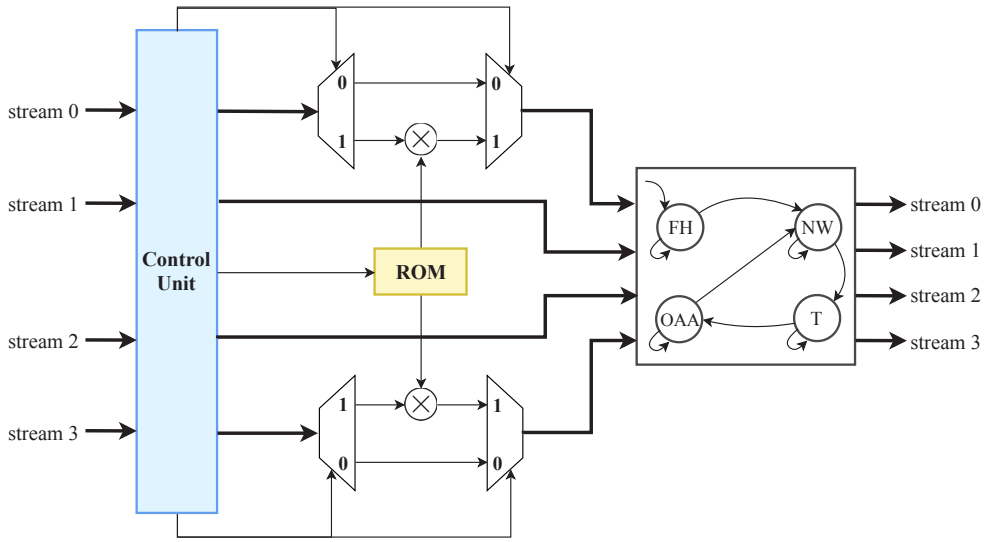


Figure 5.13: Parallel-Pipelined architecture for the WOLA module.

architecture. The size of an OFDM symbol to be transmitted is $N + L_{CP}$ and it appears at the output of the parallel-pipelined modulator in four contiguous chunks of data: stream 0 has a size of $N/4 + L_{CP}$ (1312) and the streams 1 to 3 have a size of $N/4$ (1024) each.

The implementation of dynamic frequency scaling (DFS) follows the reference design from (Tatsukawa).This design considers an FSM that reads MMCM configuration parameters pre-stored in a ROM and writes them to the DRP using the MMCM primitive. To change frequency of the output clocks, the input signal *en* must be enabled and the desired mode of operation should be given through the *mode* port. The MMCM generates a *locked* flag to indicate whether the output clocks have achieve phase and frequency alignment with the reference input clock (Figure 5.14). In this work, the DFS controller is fed by a 100 MHz reference input clock that is used to synthesize the clock signal at the output of an MMCM primitive. This synthesized clock is used for baseband processing and its frequency (f_{clkBB}) can be configured to one of four values: 20 MHz (mode 0), 40 MHz (mode 1), 80 MHz (mode 2) and 160 MHz (mode 3).

In steady-state operation, the parallel-pipelined architecture produces four samples per clock cycle. Despite this, its theoretical throughput is not limited by $4 \times f_{clkBB}$. The cyclic prefix length (L_{CP}) and the amount of non-unitary window coefficients (W) introduce a processing overhead

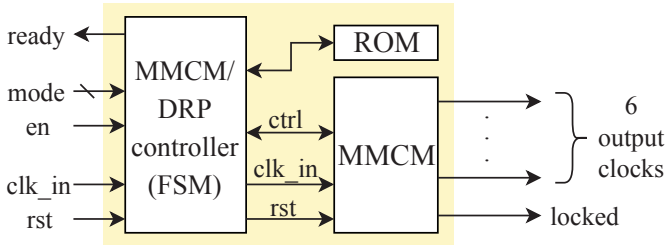


Figure 5.14: DFS controller structure (Tatsukawa)

that must be accounted for. The theoretical maximum throughput the parallel-pipelined OFDM modulator, as a function of f_{clkBB} is given by:

$$Throughput_{max} = f_{clkBB} \times \frac{N + L_{CP}}{\frac{N}{4} + L_{CP} + W} \qquad (Samples/s). \tag{5.9}$$

For the values of f_{clkBB} previously defined, we have: 65.24 MSamples/s (mode 0), 130.48 MSamples/s (mode 1), 260.95 MSamples/s (mode 2) and 521.90 MSamples/s (mode 3). These throughput values exceed the sampling frequencies defined in Table 5.11 by a safety margin.

The performance-scalable parallel-pipelined OFDM modulator was implemented and evaluated on a Xilinx ZC706 board featuring a xc7z045 Zynq device. The Zynq device is an SoC divided in two parts: a Processing System (PS) and a Programmable Logic (PL). The parallel-pipelined OFDM modulator and the DFS controller were implemented on the PL section. In turn, the PS sets up data transfers between the DDR memory (where the data to be processed is stored) and the OFDM modulator. Additionally, it issues write/read operations to the DFS controller ports through GPIO interfaces. The focus of our analysis is the parallel-pipelined OFDM modulator and to evaluate it, a simple wrapper structure was implemented in the PL section. It includes a DMA controller to speed up data transfers between the OFDM modulator and the DDR memory. It is assumed that the wrapper fetches a stream of data from the DDR and replicates it on the four parallel OFDM modulator inputs. On the modulator output side, the wrapper structure XORs the four produced output streams into a single output stream. Both input and output interfaces of the OFDM modulator are implemented as 32-bit AXI4-Stream interfaces. The simplified wrapper provides an evaluation environment for the OFDM modulator with low resource complexity and node activity, in order to reduce its share of resource usage and power consumption. The 100 MHz input clock signal for the DFS controller (FCLK0) is one of the four frequency-programmable clocks provided by the PS to the PL (Xil, b). The top-level architecture of the implemented system is shown in Figure 5.15 and its resource utilization is quantified in Table 5.12.

Although no pipelined OFDM modulator with $N = 4096$, L_{CP} and $W = 32$ was implemented in section 4.1.1, it is expected that the parallel-pipelined OFDM modulator is more resource demanding than a functionally equivalent pipelined modulator. A key factor for this observation is the replication of $N/4$-IFFT cores per parallel stream. The BRAM usage in the subcarrier mapping also registers a considerable increase due to the multi-port memory structure that implements the double-buffer. In the parallel-pipelined architecture, the increased resource utilization enables the parallelization of the modulation operations that, in turn, enables higher throughput. The resource overhead required to support DFS is almost negligible (21 slices: 70 LUTs and 79 FFs) and the latency associated with clock frequency adaptation is on average 2350 clock cycles. As the reference input clock used for DFS has a frequency of 100 MHz, this represents a latency of 23.5 µs. This latency is acceptable in this application domain, as it represents less than 1% of the 5G control plane latency requirement from (ITU-R, 2017) (10 ms).

Like in section 5.1.3, the processing throughput was calculated by a bare-metal application running on a processor. This time, it is not a soft-processor (MicroBlaze), but the ARM Cortex-9

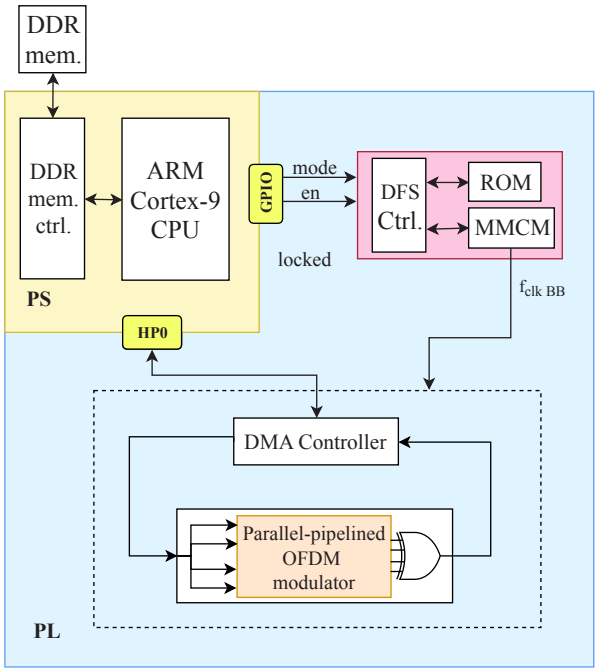


Figure 5.15: Overall system architecture

Table 5.12: Post Place-and-Route resource utilization for the performance-scalable parallel-pipelined OFDM baseband modulator. Device: xc7z045. SCM: subcarrier mapping; CPI: cyclic prefix insertion.

Resource	Available	Wrapper structure	DFS ctrl.	OFDM modulator ($f_{clkBB} = 160$ MHz)				
				SCM	IFFT	CPI	WOLA	Total for the modulator
slice	54650	1583	21	142	2248	91	1522	4070
LUT	218600	3576	70	350	5432	61	1590	7434
FF	437200	4783	79	327	6959	233	2843	11706
BRAM	545	2	0	16	39	4	0	59
DSP	900	0	0	0	76	0	2	78

Table 5.13: Parallel-pipelined OFDM modulator: processing throughput, average power consumption and energy per sample versus clock frequency

f_{clkBB} (MHz)	20	40	80	160
Throughput (MSa/s)	65.1	130.1	260.2	520.3
Avg. power (W)	0.335	0.428	0.587	0.893
Energy per sample (mJ/MSa)	5.15	3.29	2.26	1.72

hard processor available on the Zynq's PS. After an initialization latency of 4484 clock cycles, the parallel-pipelined OFDM modulator enters steady-state operation. The processing throughput for each of the four f_{clkBB} values is shown in Table 5.13. The observed throughput values are around 99.7% of the maximum theoretical threshold defined in 5.9. Beyond that, the performance requirements from Table 5.11 are satisfied, even with the modulator operating at clock frequencies that are around a third of the sampling frequencies required.

Connecting a Texas Instruments (TI) USB adapter to the PMBus port on the ZC706 board and launching the TI Fusion Digital Power Designer software on a host computer, it is possible to monitor the power rails that feed the xc7z045. The internal supply for the Zynq's PL is the VCCINT power rail and provides an nominal operating voltage of 1 V (Xil, c). The PL power consumption was measured during continuous OFDM modulator operation, that is, receiving data from DDR, processing it and sending it back to the DDR. Keeping a room temperature around 23 °C, the power measurements were repeated for the four f_{clkBB} considered: 20, 40, 80 and 160 MHz. Table 5.13 reports the obtained results.

The specialization of performance at run-time allows for an increased power efficiency. Without specialization of performance at run-time, baseband processing would be clocked at 160 MHz, in order to cover all numerologies from Table 5.11. In this scenario, the PL power consumption would settle around 0.893 W. Considering that the system has to operate in mode $\mu = 0$, the clock frequency adaptation from 160 to 20 MHz allows for a power reduction of 62.5% (558 mW). In an analogous way, the clock frequency adaptation from 160 to 40 MHz and 80 MHz reduces power consumption by 52% (465 mW) and 34.2% (306 mW), respectively. The energy per sample figures favor higher f_{clkBB} and suggest that it is more efficient to perform baseband processing as fast as possible in any situation. Despite that, in highly heterogeneous communication environments, radio devices perform very differently and low-throughput devices may not be able to follow the data rates imposed by high-throughput devices. Through performance specialization at run-time, high-throughput devices can downscale their clock frequency to reduce power consumption without compromising performance requirements and quality of service.

5.2.1 Discussion

This subsection proposed a performance-scalable, parallel-pipelined architecture for an OFDM baseband modulator implemented on a xc7z045 FPGA device. The design addresses two questions:

- How to achieve 5G-compatible baseband throughput in commercial FPGA devices?

- What is the impact run-time performance specialization in terms of resources, adaptation latency and power consumption?

The first question was addressed with the implementation of a parallel-pipelined architecture for OFDM modulation. By combining the processing of multiple samples in the same clock cycle, throughput requirements for 5G NR are achieved at relatively low clock frequencies. Regarding the second question, the specialization of performance at run-time through dynamic frequency scaling can help to handle the system's power consumption in a more efficient way and according to the communication demands. It was shown that dynamic frequency scaling has a negligible resource and latency overhead in this application domain. The results from this section and the previous one suggest that DFS and DPR can be combined into a highly flexible and efficient baseband processing system supporting the specialization of computation and performance at run-time.

5.3 Reconfigurable Baseband Modulator for Multi-mode Spectrum Aggregation

Section 1.1 states that baseband processing architectures for future wireless communications must be *flexible*, *scalable*, resource-power *efficient* and *forward-compatible*. In the two first sections of this chapter it was shown that these features can be achieved by means of DPR and/or DFS. It was also shown that the overhead introduced by these run-time reconfiguration techniques is tolerable within the context of wireless communications. In this section, DPR and DFS are combined in the same architecture to achieve the main goal of this dissertation: a dynamically reconfigurable baseband processing architecture for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation. The dual-waveform baseband modulator from section 5.1.3 already supports multiple modes and waveforms. However, the architecture comprises a single baseband processing core. Thus, it can only support spectrum aggregation of component carriers adjacent to each other - *contiguous carrier aggregation*. To enable the full potential of 5G, carrier aggregation (CA) should also be possible across separated frequency bands (Bhushan et al., 2017) - *non-contiguous CA*. For non-contiguous CA, a *multidimensional* PHY layer (and, inherently, baseband architecture) is needed, even when data aggregation schemes are not performed in the PHY layer, but in the Media Access Control (MAC) communication layer instead (Yuan et al., 2010) (Figure 5.16). Here, the attribute multidimensional means that the PHY layer is an array of independent processing blocks, rather than a monolithic structure.

The baseband architecture presented in this section features three independent modulators, whose functionality and clock frequency can be dynamically reconfigured through DPR and DFS, respectively. This setup enables the processing of multiple component carriers with different waveforms and/or baseband parameters in non-contiguous CA schemes. The architecture supports three 5G waveform candidates (OFDM, FBMC and UFMC) and, for each waveform, two modes of operation are considered. The numerologies considered in this section (Table 5.14) are a subset of those presented along Chapter 4.

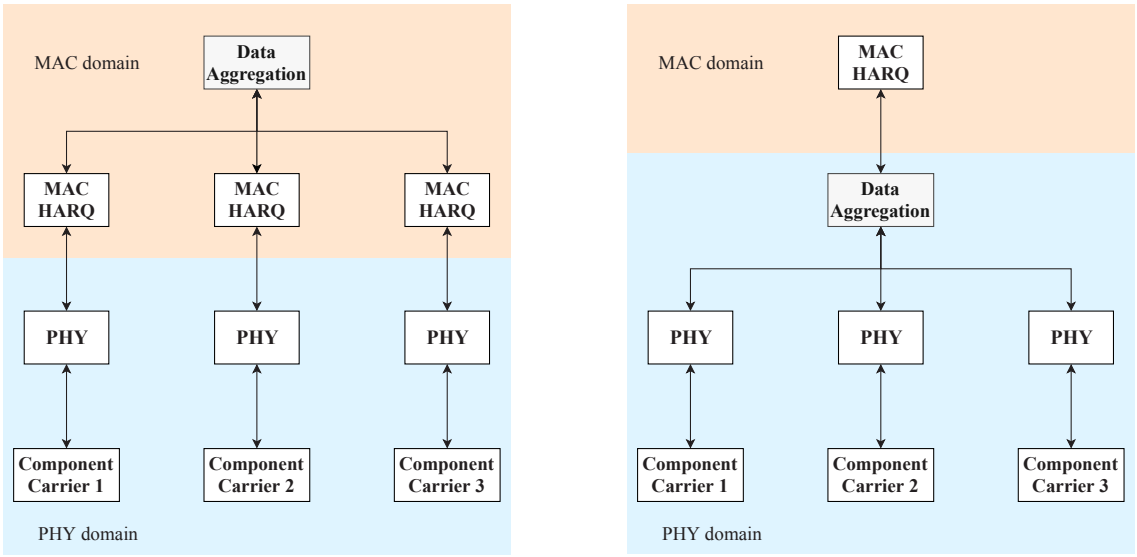


Figure 5.16: Data aggregation schemes: on the MAC layer (left) and on the PHY layer (right). HARQ: hybrid automatic repeat request entity. Schematics adapted from (Yuan et al., 2010).

Table 5.14: Waveform numerologies.

(a) OFDM

	Mode 1	Mode 2
# subcarriers, N (IFFT size)	512	1024
CP length, L_{CP}	40 (1st slot symb.) 36 (other symb.)	80 (1st slot symb.) 72 (other symb.)
WOLA samples, W	4	6

(b) FBMC

	Mode 1	Mode 2
# subcarriers, N_c	512	1024
Overlapping factor, K	4	4
IFFT size, $K.N_c$	2048	4096

(c) UFMC

	Mode 1	Mode 2
# subcarriers, N	512	1024
# subcarriers per PRB	12	12
# active PRBs	3	3
IFFT size, N'	64	64
Upsampling factor, $\frac{N}{N'}$	8	16
Filter length, L	37	73
Filter type	Dolph-Chebyshev (60-dB side lobe attenuation)	

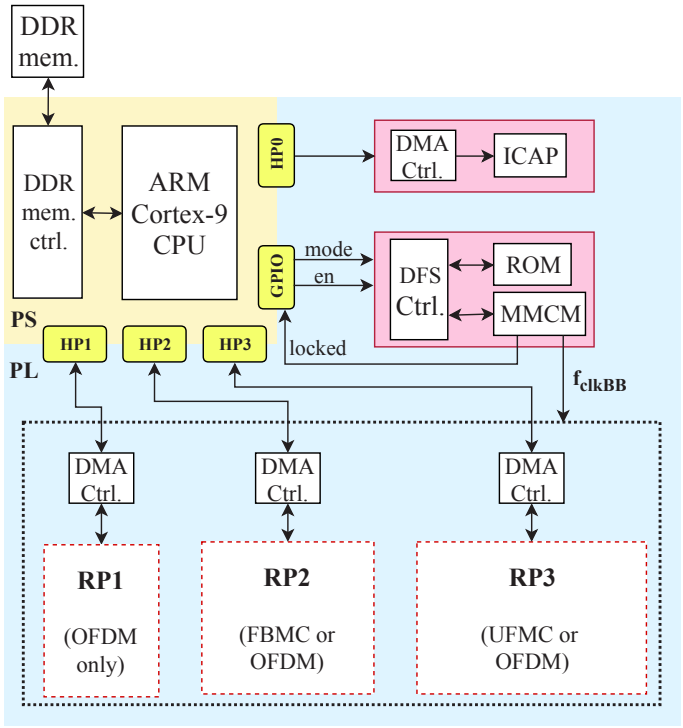


Figure 5.17: Top-level architecture for the multidimensional and reconfigurable baseband modulator. HPx: High Performance Ports, GPIO: General Purpose I/O

The multidimensional baseband modulator was implemented on an Avnet Zedboard, featuring a xc7z020 Zynq device. The system top level combines features from the designs presented in the two previous sections and can be divided in three parts. The Zedboard's 512 MB DDR memory is used as a repository for the partial bitstreams used for DPR. The Zynq's PS can be viewed as the system management unit: it is responsible for triggering the reconfiguration of the multidimensional baseband modulator and setting up data transfers between the DDR memory and the Zynq's PL. In turn, the Zynq's PL comprises the three independent baseband modulators and the infrastructure to support DPR and DFS. The modulators datapath architecture used here are those described in Chapter 4. Figure 5.17 provides a scheme for the top-level architecture of the reconfigurable, multidimensional baseband modulator

The considered architecture is inspired in the communication scenario described in (Kaltenberger et al., 2015) that combines *multi-waveform coexistence* with *dynamic spectrum access* (DSA). In this scenario, 5G communications are anchored on the pre-existing 4G infrastructure - *non-standalone 5G*. Primary 4G-LTE communications are OFDM-based and secondary 5G communications opportunistically exploit vacant spectrum resources through DSA, transmitting with different waveforms (OFDM, FBMC or UFMC). Here, the basic unit for DPR is the complete baseband datapath and each independent modulator is implemented on a reconfigurable partition (RP). From the three RPs, RP_1 is exclusively used for primary OFDM-based transmission. The two remaining RPs can be used for primary or secondary transmission: RP_2 implements FBMC or OFDM transmission modes; RP_3 implements UFMC or OFDM transmission modes. For instance, if the primary transmission

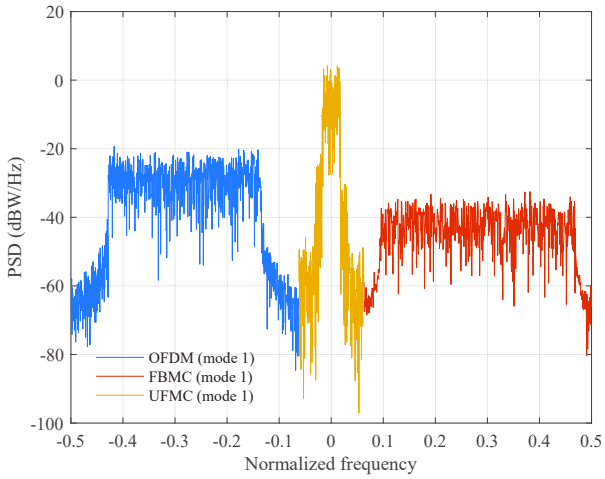


Figure 5.18: Periodograms for OFDM, FBMC and UFMC baseband signals.

requires more capacity, the three RPs can be used to independently modulate different component carriers in a non-contiguous CA scheme. On the other hand, if the primary transmission is not so demanding, RP2 and RP3 can be used for secondary *multi-waveform* 5G transmission. Figure 5.18 illustrates a potential multi-waveform coexistence scenario in a spectrum band by combining periodograms of OFDM, FBMC and UFMC baseband signals obtained from the implemented modulator datapaths.

During system initialization, the ARM core orders the downloading of partial bitstreams and input data files from an SD card to the DDR memory. For baseband processing purposes, the input data is retrieved from the DDR, sent to the baseband modulator(s) and the results are finally sent back to the DDR. Each RP has an associated DMA controller to accelerate the access to the DDR.

The DFS controller was implemented as in section 5.2, but the considered MMCM output clock frequencies are: 100 MHz, 66.7 MHz, 33.3 MHz and 16.7 MHz. The 100 MHz clock frequency was the constraint used to implement all modulator datapaths. The other values were based on the scaling of subcarrier spacing by 2^{μ}, as in 5G NR systems (TS, 2018). In this system, primary communications are based on the LTE OFDM numerologies (Table 5.14), where the subcarrier spacing (Δf) is 15 kHz. Considering OFDM mode 2 Table 5.14, the sampling frequency required is $N \times \Delta f = 15.36\,\text{MHz}$. Scaling the subcarrier spacing by 2^{μ} with $\mu = \{1,2\}$, results in sampling frequencies of $2^1 \times 15.36\,\text{MHz} = 30.72\,\text{MHz}$ and $2^2 \times 15.36\,\text{MHz} = 61.44\,\text{MHz}$. As in section 5.1.3, the ICAP is the adopted FPGA configuration interface and a dedicated DMA controller accelerates the provision of of partial bitstreams to the ICAP. The only difference is the overclocking of the ICAP at 200 MHz. The ICAP clock signal is one of the PS-generated clocks (*FCLK1*).

A general overview of the resource utilization is presented in Table 5.15. The static part occupies around 32% and 5% of the slices and BRAMs available on the xc7z020 device. Apart from PS-PL interconnect cores and DMA controllers to accelerate the baseband modulators, the static part also implements the infrastructure for reconfiguration (DPR and DFS). Like in sections 5.1.3 and 5.2,

Table 5.15: Post Place-and-Route resource utilization for the static and reconfigurable system parts.

Resource	Available	Static part (total)	Reconfig. overhead		RP_1	RP_2	RP_3	All RPs
			DFS	DPR				
Slice	13300	4210	24	424	1400	2400	3200	7000
LUT	53200	10700	75	938	5600	9600	12800	28000
FF	106400	13110	79	1292	11200	19200	25600	56000
BRAM	140	7.5	0	1.5	20	40	30	90
DSP	220	0	0	0	40	80	40	160

the resource overhead associated with DPR and DFS is small. The hardware required to implement these two reconfiguration techniques is below 2% of the available LUTs, FFs and BRAMs. The three RPs form the system's reconfigurable part and reserve 52.6%, 64.3% and 72.7% of the available slices, BRAMs and DSPs, respectively. Overall, the resource utilization for the complete system implementation represents a considerable share of the resources available in the xc7z020 device: 84.3% of slices, 69.7% of BRAMs and 72.7% of DSPs. Results for the resource utilization of each modulator datapath are presented in Table 5.16. The only significant remark comparing with the results from Chapter 4 has to do with the slice and BRAM usage in the FBMC modes. To allow for a smaller RP_2, some BRAM-based memories along the FBMC datapaths described in section 4.2 were implemented as LUT-distributed RAMs. This justifies the lower amount of BRAMs and the higher amount of slices registered for the FBMC modes in Table 5.16 compared to Table 4.6.

Table 5.16 allows for a key observation: the hardware virtualization achieved with the 7000 slices, 90 BRAMs and 160 DSPs reserved by the three RPs allows the implementation of six baseband modulators that combine for 11322 slices, 99.5 BRAMs and 106 DSPs. Adding these *virtualized* resources to the static resources exceeds the available xc7z020 slices by 17%. This is an unequivocal demonstration of the resource efficiency benefits that the application of DPR brings to multi-mode baseband processors. It is true that an equivalent static multi-mode design could benefit from the reuse of common hardware blocks between different modulator datapaths (especially between OFDM and FBMC datapaths), similarly to what was done for the static multi-mode

Table 5.16: Post Place-and-Route resource utilization for each baseband modulator datapath. Device: xc7z020; $f_{clk} = 100\,\text{MHz}$

Resource	Mode 1			Mode 2		
	OFDM	FBMC	UFMC	OFDM	FBMC	UFMC
Slice	1015	1575	2315	1126	2210	3100
LUT	2829	5103	8090	3400	7876	11782
FF	2107	2307	6279	2170	2284	9912
BRAM	7	19	11.5	10.5	40	11.5
DSP	14	21	18	14	21	18

dual-waveform modulator from section 5.1.3. Still, implementing the multidimensional baseband modulator as a static multi-mode design would be challenging given the resource budget available on cost-optimized devices, like the xc7z020. An immediate counterargument is the existence of a wide range of FPGA/SoC devices with larger area and logic density. However, this would decrease the system's cost-effectiveness: an FPGA with a larger chip area is more expensive and likely to consume more power (Vipin and Fahmy, 2018). Considering the modes of operation from Table 5.14 and that all 3 RPs are in use, the proposed design supports 32 combinations of baseband modulators: $2\ RP_1\ modes \times 4\ RP_2\ modes \times 4\ RP_3\ modes$. Yet, the application of DPR eases the system upgrade with new modes of operation in order to extend the system duty lifetime. The addition of modes of operation is not limited by the available resources on the FPGA device, but instead by the resources reserved by the RPs and the capacity to store partial bitstreams (512 MB DDR memory, in this case). During the DPR design flow using Xilinx Vivado EDA tool, the different system configurations are created from a design checkpoint that saves the floorplaning and routing of the system's static part, leaving the RPs as empty *black boxes*. New configurations can be created by designing new circuit configurations for these black boxes and generating the corresponding partial bitstreams. This design reusability makes the system adaptable and reduces the upgrade design time.

The Zedboard has no embedded circuit setup or PMBus port to measure power consumption. So, the dynamic power consumption for each modulator datapath and baseband clock frequency was estimated by the Power Analysis tool from Vivado 2015.2. The high-confidence estimates were performed using placed and routed netlists and node activity files. The results are presented in Table 5.17. The UFMC modulator modes have a higher dynamic power consumption compared to FBMC and OFDM. This is mainly due to the higher resource usage and node activity involved in UFMC datapaths. The clock frequency adaptation allowed by DFS results in power savings that tend to be more evident for the most resource-demanding modes of operation (UFMC and FBMC). Compared to a design with baseband clock frequency fixed to 100 MHz:

- the clock frequency adaptation to 66.7 MHz results in dynamic power savings between 39 mW (35% reduction in OFDM mode 1) and 82 mW (51% reduction in FBMC mode 2);
- the clock frequency adaptation to 33.3 MHz results in dynamic power savings between 79 mW (70% reduction in OFDM mode 1) and 156 mW (67% reduction in UFMC mode 2);
- the clock frequency adaptation to 16.7 MHz results in dynamic power savings between 99 mW (88% reduction in OFDM mode 1) and 194 mW (83% reduction in UFMC mode 2);

For the set of baseband clock frequencies defined, the DFS procedure took on average 47 µs to modify and lock the MMCM output clock. Despite higher than the latency registered in section 5.1, this latency is still acceptable in 5G NR communications. This analysis reinforces the results from the previous section: the performance specialization at run-time improves the system power efficiency with negligible resource and latency overhead.

In the multidimensional baseband modulator, the area and amount of RP reserved resources are higher than in the previously presented designs, resulting in larger bitstream sizes. On the other

Table 5.17: Dynamic power consumption estimates for the six implemented baseband modulator cores (in mW). Device: xc7z020; Analysis tool: Vivado 2015.2; Post Place-and-Route power analysis with high confidence level; Node activity derived from Post Place-and-Route simulation.

f_{clk}	Mode 1			Mode 2		
	OFDM	FBMC	UFMC	OFDM	FBMC	UFMC
100 MHz	113	148	180	123	161	233
66.7 MHz	74	84	119	78	79	155
33.3 MHz	34	25	60	33	28	77
16.7 MHz	14	8	30	10	10	39

hand, the reconfiguration speed was increased through ICAP overclocking. Table 5.18 quantifies the DPR latency and compressed bitstream sizes for the worst-case scenario in each RP. The largest RP (RP_3) takes up to 767 µs to be reconfigured, corresponding to the transfer of a 596 kB bitstream to the ICAP. Throughout all DPR latency measurements, the reconfiguration throughput was at least 790 MB/s. This value is about 99% of the theoretical ICAP throughput, considering 32-bit transfers and overclocking at 200 MHz. In general terms, the DPR latency for each individual RP is below 1 ms, while the overall reconfiguration of the three RPs takes less than 2 ms.

Again, these latencies are within an acceptable range considering the control plane requirements from (ITU-R, 2017). However, the ITU report from (ITU-R, 2017) also states that in critical, ultra low-latency scenarios, a *make-before-break* approach must be adopted to completely mitigate the control plane latency. In other words, the control plane latency must be *hidden* by setting up a new communication channel before breaking the current one. Under these circumstances, a high-priority communication can reserve a spare RP to seamlessly adapt the transmission mode. This scenario is exemplified in Figure 5.19. Let us assume that RP_1 is currently performing baseband modulation for a ultra low-latency communication. This transmission needs to be adapted from OFDM mode 1 to OFDM mode 2, without breaking the current communication link. RP_2 is currently unused and is reconfigured to OFDM mode 2 before baseband processing at RP_1 terminates. The ultra low-latency communication commutes its baseband processing from RP_1 to RP_2, adapting its baseband processing operation without any DPR latency penalty.

5.3.1 Discussion

This section presents a reconfigurable, multidimensional baseband modulator architecture suitable for multi-mode, multiple waveform coexistence and dynamic spectrum aggregation scenarios. The

Table 5.18: DPR latency (worst-case scenarios)

	RP_1	RP_2	RP_3
DPR latency	400 µs	677 µs	767 µs
Partial bitstream size	309 kB	526 kB	596 kB

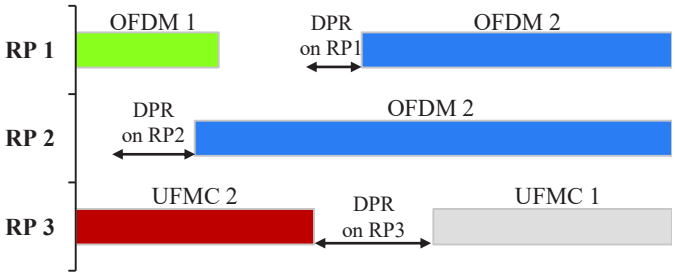


Figure 5.19: Example of *make-before-break* approach to mitigate DPR latency

design builds upon the results from the two previous sections to combine the run-time specialization of computation and performance. By featuring three independent and reconfigurable baseband modulators, the architecture allows the processing of up to three component carriers using different waveforms (OFDM, FBMC and UFMC) and/or numerologies. Although, the total reconfigurable area of the system covers more than half the available xc7z020 resources, the ICAP overclocking contributes to maintain the DPR latency under tolerable values. In this design, the performance specialization through DFS resulted in dynamic power savings up to 194 mHz. Besides flexibility, scalability and forward-compatibility, the *cost-effectiveness* is perhaps the most prominent feature of this architecture. It is clearly demonstrated how the hardware virtualization through DPR enables implementations that exceed the hardware resources available on an FPGA device. This allows the system implementation on a small-form, cost-optimized device with immediate cost and power consumption benefits and without compromising the system functionality.

5.4 Summary

This chapter started with the study of the DPR application to design dynamically reconfigurable architectures for baseband processing. After testing several strategies to define and dimension reconfigurable partitions, a DPR-based dual-waveform modulator was compared to an equivalent static multi-mode design. The DPR-design was shown to be superior in terms of performance, resource utilization, functional density and power consumption. Moreover, the DPR overhead in terms of resources, reconfiguration latency and energy was measured. The results validate the hypothesis formulated in the beginning of this dissertation: DPR is a convenient technique to design flexible, reconfigurable and efficient baseband processors for future wireless communications.

Next, an OFDM modulator supporting the wide range of 5G NR throughput requirements was proposed. The family of 5G NR numerologies imposes sampling frequencies ranging from tens of MHz to about 500 MHz. Featuring a parallel-pipelined architecture, the modulator cooperatively processes multiple samples in a single clock cycle and is able to produce more than 500 MSamples/s at a 160 MHz clock frequency. The specialization of performance at run-time is evaluated in this design. Through DFS, the modulator performance can be dynamically scaled to cover different 5G NR numerologies in a power-efficient way and with a negligible resource and latency overhead.

Finally, a multidimensional baseband modulator architecture is presented. It comprises three independent baseband modulators whose waveform scheme can be adapted through DPR and whose performance can be scaled through DFS. This makes the architecture suitable for multi-waveform and dynamic spectrum aggregation scenarios. The proposed architecture targets the main goal of this dissertation and presents the features required in baseband processing infrastructures for future wireless: it is flexible, scalable, cost-effective and forward-compatible.

Chapter 6

Conclusions

More than a technological advancement, the next generation of wireless communications represents a revolution in the way humans and machines interact with each other. The realization of this revolution requires considerable changes in the existing wireless infrastructure. In particular, the communication physical layer must be extremely flexible to support an unprecedented range of use cases and requirements. Considering FPGAs as convenient platforms to design hardware infrastructures for future wireless devices, this dissertation investigates a dynamically reconfigurable baseband processing architecture for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation. In the proposed architecture, *dynamic partial reconfiguration* (DPR) and *dynamic frequency scaling* (DFS) are evaluated as appropriate architectural concepts to produce flexible, scalable, efficient and forward-compatible baseband processors.

6.1 Work Outcomes

The adopted approach starts with the study of three 5G waveform candidates (OFDM, FBMC and UFMC) in order to identify scenarios where the application of run-time reconfiguration to the baseband processing infrastructure is advantageous. The signal synthesis and analysis for these waveforms involve IFFT/FFT operations, which represent the bulk of computation complexity within the baseband processing datapath. For this reason, Chapter 3 studies hardware implementation for the two main types of FFT architectures: *pipelined* and *memory-based*. Pipeline architectures are preferable for OFDM and FBMC processing, and can benefit from dynamic reconfiguration due to considerable resource usage and power consumption variation across different FFT sizes. In sparse-data scenarios, the bypass of zero-value multiplications did not result in consistent dynamic power savings. The memory-based FFT architecture presented in this chapter was shown to be suitable for UFMC modulation. Despite its lower processing throughput, a memory-based FFT core can be replicated for each UFMC subband without considerable resource and power penalties. After the study of hardware implementation for FFT cores, this dissertation proceeded with the implementation of baseband processors for OFDM modulation and demodulation, FBMC and UFMC modulation - Chapter 4. For each case, datapaths for different numerologies were

implemented in order to study the impact of baseband parameter variation on each datapath module. The results encouraged the application of run-time circuit specialization, especially for the waveform modulators. Chapter 4 advances the state of the art with the first published FPGA-based implementations of an FS-FBMC-OQAM baseband modulator. The design is compared to published PPN-FBMC modulators and the results suggest that the FS-FBMC-OQAM modulation uses less resources than an equivalent PPN-FBMC design, considering FPGA implementations.

In Chapter 5, DPR and DFS are applied to the baseband modulators from Chapter 4. Besides the application of run-time reconfiguration techniques, their benefits are weighted against their overheads. The study on a dual-waveform baseband modulator contributes an extensive and fair comparison between a DPR-based design and a functional equivalent static multi-mode design. The comparative analysis concluded that a DPR-based design outperforms the static multi-mode design in terms of resource utilization, functional density and power consumption. Real-time measurements also showed that the DPR latency and energy overhead are acceptable in the context of modern and future wireless communications, especially for commercial base stations. Chapter 5 also presented a parallel-pipelined OFDM modulator that can achieve high processing throughput with low clock frequency. DFS is applied to this design, allowing the run-time performance specialization to cover the range of 5G New Radio requirements. The on-the-fly clock frequency adaptation improves the system power efficiency with a negligible resource and latency overhead.

Having shown the viability of both DPR and DFS to design reconfigurable baseband processors, the final contribution of Chapter 5 is a baseband modulator architecture for multi-mode transmission, multi-waveform coexistence and dynamic spectrum aggregation scenarios. This architecture has three independent modulators whose waveform and mode of operation can be reconfigured at run-time through DPR. This allows the processing of up to three component carriers in non-contiguous spectrum aggregation schemes. The clock frequency for baseband processing is adapted through DFS to scale the performance and avoid power over-consumption. Taking advantage of the hardware virtualization enabled by DPR, the overall circuitry implemented in this architecture exceeds the resources available on the used FPGA device. This allows the system implementation on a small-form, cost-effective FPGA device, which cheaper and less power-hungry. The presented design is an unequivocal evidence of the DPR potential to enable flexible, scalable, cost-effective and forward-compatible baseband processing infrastructures for future wireless.

Apart from multi-waveform coexistence and dynamic spectrum aggregation, the reconfigurable baseband architectures proposed in this dissertation also find a suitable application in Cloud Radio Access Networks (C-RANs) (Wu et al., 2015). This type of network architecture is viewed as a 5G key enabler (Andrews et al., 2014) and attempts to achieve a more efficient resource allocation between radio accesses by considering a central baseband processing unit (BBU) that serves multiple, distributed Remote Radio Heads (RRH). The centralization of baseband processing allows for increased energy and spectral efficiency, but requires the implementation of BBUs as pools of baseband processing resources that are dynamically assigned to different tasks and should support the different access technologies and modes of operation used by RRHs. The high versatility of architecture proposed in this dissertation could be exploited in C-RAN BBUs: the

independent baseband modulators could be dynamically reconfigured to support different radio access technologies according to the communication demands. In the face of the emergence of new radio access technologies or modes of operation, the proposed architecture can be easily upgraded without requiring the complete redesign of the baseband infrastructure.

6.2 Future Work

A possible agenda to further extend the investigation present in this dissertation research would consider two main directions: *1)* the study and development of a context-aware management unit capable of intelligently and efficiently adapt the proposed reconfigurable baseband processing architecture; *2)* the integration of the proposed baseband processing architectures into a Proof of Concept (PoC) prototype for a flexible and adaptable radio system.

Reconfiguration Management Unit The management unit would be responsible for controlling and adapting the baseband operation according to the communication scenario. This would require the implementation of a *cognitive cycle*. Although there are several notions of the cognitive cycle (Mitola and Maguire, 1999; Haykin, 2005), it can be divided in four stages: *observe*, *predict*, *decide* and *act*. During the *observe* stage, context-awareness is built upon the information about external stimuli provided by the upper layers of the communication system, as well as information about the current baseband processing configuration. This information may relate to electromagnetic stimuli (e.g.: channel quality, spectrum utilization), device operation constraints (e.g.: power consumption, temperature) or communication constraints (e.g.: wireless standard, data rate, MIMO transmission). Based on the current system state and on the output from the *observe* stage, the management unit *predicts* upcoming events or convenient operational states. At this stage, *machine learning* could be employed (Rihani et al., 2018) to learn from previous events and improve the prediction quality. Both observations and predictions would be used to *decide* on what/how/when to reconfigure the radio system and *act* accordingly. The evolution of FPGA devices towards SoC results in heterogeneous HW/SW systems with possibly conflicting mixed criticalities. Thus, the management unit could benefit from hypervisors for dynamic reconfigurable systems, such as the one proposed in (Janßen et al., 2017), in order to enable the parallelization between system management and baseband processing tasks. Ultimately, the tight integration of the management unit with the baseband processing infrastructure would result in a self-adaptive, highly efficient and responsive system that could have an overall positive impact on the communication quality of service.

PoC for a Flexible and Adaptable Radio System A first step towards a flexible and agile radio system would be the integration with FPGA-based All-digital Transceivers (ADTs) (Prata et al., 2016). The ADT concept moves the radio digital domain closer to the antenna and its implementation on FPGAs extends flexibility and real-time reconfigurability to most of the radio physical layer. Another important aspect is the extension of run-time reconfigurability to FEC coders/decoding.

This requires the study of error-correcting codes proposed for 5G New Radio, such as low-density parity-check (LDPC) and polar codes (Hui et al., 2018), and their efficient implementation on FPGA. Extension of the analysis from Chapter 4 to FBMC and UFMC demodulators or even to other 5G candidate waveforms, such as GFDM (Nimr et al., 2018) are also potential future developments towards the implementation of a flexible and adaptable Radio System.

6.3 Final Remarks

This work presented a dynamically reconfigurable baseband processing architecture for multi-mode, multi-waveform coexistence and dynamic spectrum aggregation. The architecture is FPGA-oriented and explores the potential of *dynamic partial reconfiguration* and *dynamic frequency scaling*. These run-time reconfiguration techniques are shown to be viable and in the domain of wireless communications and turn the system flexible, scalable, efficient and forward-compatible.

The baseband processor can adapt its mode of operation and performance at run-time to support different communication setups and satisfy the current communication demands. Moreover, the specialization of computation and performance at run-time improve the overall resource ad power efficiency. Through hardware virtualization, the proposed architecture can be implemented on cost-effective devices and its set of functionalities can be easily extended to support communication setups needed in the future.

The involvement and commitment by FPGA vendors to the development of solutions for the main 5G challenges clearly shows the a major role that Reconfigurable Computing will play in the realization of future wireless communications. This dissertation is a contribution to the field and can find particular application on centralized and heterogeneous baseband processing units in Cloud-RANs.

References

Fusion Digital Power Designer | TI.com. http://www.ti.com/tool/fusion_digital_power_designer. Accessed: 2017-09-08.

Cyclone v SoC FPGAs- Intel SoC FPGA. URL https://www.intel.com/content/www/us/en/products/programmable/soc/cyclone-v.html. Accessed: 2018-12-07.

OFDM demodulation - MATLAB lteOFDMDemodulate. https://www.mathworks.com/help/lte/ref/lteofdmdemodulate.html, a. Accessed: 2016-05-05.

OFDM modulation - MATLAB lteOFDMModulate. http://www.mathworks.com/help/lte/ref/lteofdmmodulate.html, b. Accessed: 2016-05-05.

FBMC vs. OFDM Modulation - MATLAB & Simulink Example. https://www.mathworks.com/help/comm/examples/fbmc-vs-ofdm-modulation.html, a. Accessed: 2018-08-08.

UFMC vs. OFDM Modulation - MATLAB & Simulink Example. https://www.mathworks.com/help/comm/examples/ufmc-vs-ofdm-modulation.html, b. Accessed: 2018-08-08.

Ettus Research - High performance Software Defined Radio (SDR). https://www.ettus.com/product/category/USRP-X-Series. Accessed: 2018-12-07.

WARP Project. http://warpproject.org. Accessed: 21/08/2015.

SoC with Hardware and Software Programmability. URL https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Accessed: 2018-12-07.

Sassan Ahmadi. Towards 5G: Xilinx Solutions and Enablers for Next-Generation Wireless Systems. Technical Report WP476, Xilinx Inc., Jun 2016. v1.0.

R. Airoldi, O. Anjum, F. Garzia, A. Wyglinski, and J. Nurmi. Energy-Efficient Fast Fourier Transforms for Cognitive Radio Systems. *IEEE Micro*, 30(6):66–76, Nov 2010. ISSN 0272-1732. doi: 10.1109/MM.2010.84.

Roberto Airoldi, Fabio Campi, Manuele Cucchi, Deepak Revanna, Omer Anjum, and Jari Nurmi. Design and Implementation of a Power-aware FFT Core for OFDM-based DSA-enabled Cognitive Radios. *Journal of Signal Processing Systems*, 78(3):257–265, Mar 2015. ISSN 1939-8115. doi: 10.1007/s11265-014-0894-z.

Rami Akeela and Behnam Dezfouli. Software-defined Radios: Architecture, state-of-the-art, and challenges. *Computer Communications*, 128:106–125, September 2018. ISSN 0140-3664. doi: 10.1016/j.comcom.2018.07.012.

Ian F. Akyildiz, Shuai Nie, Shih-Chun Lin, and Manoj Chandrasekaran. 5G roadmap: 10 key enabling technologies. *Computer Networks*, 106:17 – 48, 2016. ISSN 1389-1286. doi: 10.1016/j.comnet.2016.06.010.

R. G. Alves, P. L. Osorio, and M. N. S. Swamy. General FFT pruning algorithm. In *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, volume 3, pages 1192–1195, Aug 2000. doi: 10.1109/MWSCAS.2000.951428.

J.G. Andrews, S. Buzzi, Wan Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, and J.C. Zhang. What Will 5G Be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, June 2014. ISSN 0733-8716. doi: 10.1109/JSAC.2014.2328098.

M. Bellanger. FS-FBMC: An alternative scheme for filter bank based multicarrier transmission. In *2012 5th International Symposium on Communications, Control and Signal Processing*, May 2012.

M. Bellanger, D. Le Ruyet, D. Roviras, M. Terré, J. Nossek, L. Baltar, Q. Bai, D. Waldhauser, M. Renfors, T. Ihalainen, et al. FBMC physical layer: a primer. Technical report, PHYDYAS Project, 2010.

Vincent Berg, Jean-Baptiste Doré, and Dominique Noguet. A flexible radio transceiver for TVWS based on FBMC. *Microprocessors and Microsystems*, 38(8, Part A):743 – 753, 2014. ISSN 0141-9331. doi: https://doi.org/10.1016/j.micpro.2014.05.010. 2013 edition of the Euromicro Conference on Digital System Design (DSD 2013).

N. Bhushan, T. Ji, O. Koymen, J. Smee, J. Soriaga, S. Subramanian, and Y. Wei. Industry Perspective - 5G Air Interface System Design Principles. *IEEE Wireless Communications*, 24(5):6–8, October 2017. ISSN 1536-1284. doi: 10.1109/MWC.2017.8088414.

J. A. C. Bingham. Multicarrier modulation for data transmission: an idea whose time has come. *IEEE Communications Magazine*, 28(5):5–14, May 1990. ISSN 0163-6804. doi: 10.1109/35.54342.

M. Boban, K. Manolakis, M. Ibrahim, S. Bazzi, and W. Xu. Design aspects for 5G V2X physical layer. In *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–7, Oct 2016. doi: 10.1109/CSCN.2016.7785161.

H. Bogucka, P. Kryszkiewicz, and A. Kliks. Dynamic spectrum aggregation for future 5G communications. *IEEE Communications Magazine*, 53(5):35–43, May 2015. ISSN 0163-6804. doi: 10.1109/MCOM.2015.7105639.

R. Bonamy, S. Bilavarn, D. Chillet, and O. Sentieys. Power consumption models for the use of dynamic and partial reconfiguration. *Microprocessors and Microsystems*, 38(8-B):860 – 872, 2014. ISSN 0141-9331.

Karen Campbell, Jim Diffley, Bob Flanagan, Bill Morelli, Brendan O'Neil, and Francis Sideco. The 5G economy: How 5G technology will contribute to the global economy. 2017.

M. Carvalho, M. L. Ferreira, and J. C. Ferreira. FPGA-based implementation of a frequency spreading FBMC-OQAM baseband modulator. In *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 174–177, Dec 2017. doi: 10.1109/ICECS.2017.8292011.

Miguel Carvalho. FPGA Implementation of a Baseband Processor for FBMC Transmission. MSc Thesis, Faculty of Engineering of the University of Porto, July 2017.

J. Chacko, C. Sahin, D. Nguyen, D. Pfeil, N. Kandasamy, and K. Dandekar. FPGA-based latency-insensitive OFDM pipeline for wireless research. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, Sept 2014. doi: 10.1109/HPEC.2014.7040969.

D. Chang. Effect and Compensation of Symbol Timing Offset in OFDM Systems With Channel Interpolation. *IEEE Transactions on Broadcasting*, 54(4):761–770, Dec 2008. ISSN 0018-9316. doi: 10.1109/TBC.2008.2002339.

E. Chen and C. Chu. Channel estimation for NC-OFDM systems based on subspace pursuit algorithm. In *2012 IEEE 11th International Conference on Signal Processing*, volume 1, pages 88–91, Oct 2012. doi: 10.1109/ICoSP.2012.6491607.

Inkeun Cho, T. Patyk, D. Guevorkian, J. Takala, and S. Bhattacharyya. Pipelined FFT for wireless communications supporting 128-2048 / 1536 -point transforms. In *2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1242–1245, December 2013. doi: 10.1109/GlobalSIP.2013.6737133.

M. Dalla Cia, F. Mason, D. Peron, F. Chiariotti, M. Polese, T. Mahmoodi, M. Zorzi, and A. Zanella. Using Smart City Data in 5G Self-Organizing Networks. *IEEE Internet of Things Journal*, 5(2): 645–654, April 2018. ISSN 2327-4662. doi: 10.1109/JIOT.2017.2752761.

Christopher Claus, Rehan Ahmed, Florian Altenried, and Walter Stechele. Towards Rapid Dynamic Partial Reconfiguration in Video-Based Driver Assistance Systems. In Phaophak Sirisuk, Fearghal Morgan, Tarek El-Ghazawi, and Hideharu Amano, editors, *Applied Reconfigurable Computing: Architectures, Tools and Applications*, pages 55–67. Springer Berlin Heidelberg, 2010.

James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301, 1965. ISSN 0025-5718. doi: 10.2307/2003354.

A. Cortés, I. Vélez, I. Zalbide, A. Irizar, and J. F. Sevillano. An FFT Core for DVB-T/DVB-H Receivers. *VLSI Des.*, 2008(2):12:1–12:9, January 2008. ISSN 1065-514X. doi: 10.1155/2008/610420.

L.H. Crockett, R.A. Elliot, and M.A. Enderwitz. *The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC*. Strathclyde Academic Media, 2014. ISBN 9780992978709.

J.-P. Delahaye, Jacques Palicot, C. Moy, and P. Leray. Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform. In *16th IST Mobile and Wireless Communications Summit*, July 2007. doi: 10.1109/ISTMWC.2007.4299250.

J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and Jacques Palicot. A FPGA partial reconfiguration design approach for cognitive radio based on NoC architecture. In *2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008.*, pages 355–358, June 2008. doi: 10.1109/NEWCAS.2008.4606394.

D. C. Dinis, R. F. Cordeiro, F. M. Barradas, A. S. R. Oliveira, and J. Vieira. Agile Single- and Dual-Band All-Digital Transmitter Based on a Precompensated Tunable Delta-Sigma Modulator. *IEEE Transactions on Microwave Theory and Techniques*, 64(12):4720–4730, Dec 2016. ISSN 0018-9480. doi: 10.1109/TMTT.2016.2622696.

Jean-Baptiste Doré, Robin Gerzaguet, Nicolas Cassiau, and Dimitri Ktenas. Waveform contenders for 5G: Description, analysis and comparison. 24:46–61. ISSN 1874-4907. doi: 10.1016/j.phycom.2017.05.004.

B. Drozdenko, M. Zimmermann, T. Dao, K. Chowdhury, and M. Leeser. Hardware-Software Codesign of Wireless Transceivers on Zynq Heterogeneous Systems. *IEEE Transactions on Emerging Topics in Computing*, 6(4):566–578, Oct 2018. ISSN 2168-6750. doi: 10.1109/TETC.2017.2651054.

A. Dutta, D. Saha, D. Grunwald, and D. Sicker. An architecture for Software Defined Cognitive Radio. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2010*, pages 1–12, Oct 2010a.

Aveek Dutta, Dola Saha, Dirk Grunwald, and Douglas Sicker. Practical Implementation of Blind Synchronization in NC-OFDM Based Cognitive Radio Networks. In *Proceedings of the 2010 ACM Workshop on Cognitive Radio Networks*, CoRoNet '10, pages 1–6, New York, NY, USA, 2010b. ACM. ISBN 978-1-4503-0141-1. doi: 10.1145/1859955.1859957.

B. Farhang-Boroujeny. OFDM Versus Filter Bank Multicarrier. *IEEE Signal Processing Magazine*, 28(3):92–112, May 2011. ISSN 1053-5888. doi: 10.1109/MSP.2011.940267.

M. L. Ferreira and J. C. Ferreira. Flexible and Dynamically Reconfigurable FPGA-Based FS-FBMC Baseband Modulator. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018. doi: 10.1109/ISCAS.2018.8351060.

M. L. Ferreira, A. Barahimi, and J. C. Ferreira. Dynamically reconfigurable LTE-compliant OFDM modulator for downlink transmission. In *2016 Conference on Design of Circuits and Integrated Systems*, Nov 2016. doi: 10.1109/DCIS.2016.7845359.

Mário Lopes Ferreira, João Canas Ferreira, and Michael Hübner. A Parallel-Pipelined OFDM Baseband Modulator with Dynamic Frequency Scaling for 5G Systems. In Nikolaos Voros, Michael Huebner, Georgios Keramidas, Diana Goehringer, Christos Antonopoulos, and Pedro C. Diniz, editors, *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, pages 511–522. Springer International Publishing, 2018. ISBN 978-3-319-78890-6.

M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson. Pipelined Radix-2^k Feedforward FFT Architectures. *IEEE Transactions on Very Large Scale Integration Systems*, 21(1):23–32, Jan 2013. ISSN 1063-8210.

Amitabha Ghosh and Rapeepat Ratasuk. *Essentials of LTE and LTE-A*. The Cambridge Wireless Essentials Series. Cambridge University Press, 2011. doi: 10.1017/CBO9780511997082.

I.J. Good. The Relationship Between Two Fast Fourier Transforms. *IEEE Transactions on Computers*, C-20(3):310–317, March 1971. ISSN 0018-9340. doi: 10.1109/T-C.1971.223236.

H. Gu and S. Chen. Partial Reconfiguration Bitstream Compression for Virtex FPGAs. In *2008 Congress on Image and Signal Processing*, volume 5, pages 183–185, May 2008. doi: 10.1109/CISP.2008.253.

S. G. Hansen, D. Koch, and J. Torresen. High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 174–180, May 2011. doi: 10.1109/IPDPS.2011.139.

S. Haykin. Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, 23(2):201–220, Feb 2005. ISSN 0733-8716. doi: 10.1109/JSAC.2004.839380.

Ke He, Louise Crockett, and Robert Stewart. Dynamic Reconfiguration Technologies Based on FPGA in Software Defined Radio System. *Journal of Signal Processing Systems*, 69(1):75–85, 2011. ISSN 1939-8115.

S. He and M. Torkelson. A new approach to pipeline FFT processor. In *Proceedings of IPPS '96, The 10th International Parallel Processing Symposium, 1996.*, pages 766–770, Apr 1996. doi: 10.1109/IPPS.1996.508145.

S. He and M. Torkelson. Design and implementation of a 1024-point pipeline FFT processor. In *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pages 131–134, May 1998. doi: 10.1109/CICC.1998.694922.

J.A. Hidalgo, J. Lopez, F. Arguello, and E.L. Zapata. Area-efficient architecture for Fast Fourier transform. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(2):187–193, Feb 1999. ISSN 1057-7130. doi: 10.1109/82.752951.

Pao-Ann Hsiung, Marco D. Santambrogio, and Chun-Hsian Huang. *Reconfigurable System Design and Verification*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2009. ISBN 1420062662, 9781420062663.

D. Hui, S. Sandberg, Y. Blankenship, M. Andersson, and L. Grosjean. Channel Coding in 5G New Radio: A Tutorial Overview and Performance Comparison with 4G LTE. *IEEE Vehicular Technology Magazine*, 13(4):60–69, Dec 2018. ISSN 1556-6072. doi: 10.1109/MVT.2018.2867640.

T. Hwang, C. Yang, G. Wu, S. Li, and G. Y. Li. OFDM and Its Wireless Applications: A Survey. *IEEE Transactions on Vehicular Technology*, 58(4):1673–1694, May 2009. ISSN 0018-9545. doi: 10.1109/TVT.2008.2004555.

T. Ihalainen, A. Viholainen, T. H. Stitz, and M. Renfors. Generation of filter bank-based multicarrier waveform using partial synthesis and time domain interpolation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(7):1767–1778, July 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2009.2034237.

ITU-R. IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond. Technical report, ITU-R, September 2015. ITU-R M.2083-0.

ITU-R. Minimum requirements related to technical performance for IMT-2020 radio interface(s). Technical Report M.2410-0, ITU-R, November 2017. URL https://www.itu.int/pub/R-REP-M.2410-2017.

Atif Raza Jafri, Javaria Majid, Lei Zhang, Muhammad Ali Imran, and M. Najam ul islam. FPGA Implementation of UFMC based baseband transmitter: case study for LTE 10MHz channelization. *Wireless Communications and Mobile Computing*, June 2018.

I. Jang, Z. Piao, Z. Dong, J. Chung, and K. Lee. Low-power FFT design for NC-OFDM in cognitive radio systems. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2449–2452, May 2011. doi: 10.1109/ISCAS.2011.5938099.

B. Janßen, F. Korkmaz, H. Derya, M. Hübner, M. L. Ferreira, and J. C. Ferreira. Towards a type 0 hypervisor for dynamic reconfigurable systems. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–7, Dec 2017. doi: 10.1109/RECONFIG.2017.8279825.

Greg Jue. Exploring 5G Coexistence Scenarios Using a Flexible Hardware/Software Testbed - Application Note, December 2017.

Florian Kaltenberger, Raymond Knopp, Carmine Vitiello, Martin Danneberg, and Andreas Festag. Experimental analysis of 5G candidate waveforms and their coexistence with 4G systems. In *JNCW 2015, Joint NEWCOM/COST Workshop on Wireless Communications*, Oct 2015.

Tarik Kazaz, Christophe Van Praet, Merima Kulin, Pieter Willemen, and Ingrid Moerman. Hardware accelerated SDR platform for adaptive air interfaces. In *ETSI Workshop on Future Radio Technologies: Air Interfaces*, 2016.

M. G. Kibria, G. P. Villardi, K. Ishizu, and F. Kojima. Throughput Enhancement of Multicarrier Cognitive M2M Networks: Universal-Filtered OFDM Systems. *IEEE Internet of Things Journal*, 3(5):830–838, Oct 2016. ISSN 2327-4662. doi: 10.1109/JIOT.2015.2509259.

Raymond Knopp, Florian Kaltenberger, Carmine Vitiello, and Marco Luise. Universal filtered multicarrier for machine type communications in 5G. In *EUCNC 2016, European Conference on Networks and Communications*, June 2016.

Charles Eric LaForest and J. Gregory Steffan. Efficient Multi-ported Memories for FPGAs. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 41–50, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-911-4. doi: 10.1145/1723112.1723122.

K. Li, M. Wu, G. Wang, and J. R. Cavallaro. A high performance GPU-based software-defined basestation. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, pages 2060–2064, Nov 2014. doi: 10.1109/ACSSC.2014.7094835.

S. Lien, S. Shieh, Y. Huang, B. Su, Y. Hsu, and H. Wei. 5G New Radio: Waveform, Frame Structure, Multiple Access, and Initial Access. *IEEE Communications Magazine*, 55(6):64–71, 2017. ISSN 0163-6804. doi: 10.1109/MCOM.2017.1601107.

S. J. Lin and W. H. Chung. The split-radix fast Fourier transforms with radix-4 butterfly units. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, Oct 2013.

Shaoshan Liu, Richard Neil Pittman, and Alessandro Forin. Energy Reduction with Run-Time Partial Reconfiguration. Technical Report MSR-TR-2009-2017, Sep 2009.

J. Löfgren and P. Nilsson. On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems. In *NORCHIP, 2011*, pages 1–4, Nov 2011. doi: 10.1109/NORCHP.2011.6126703.

Mário Lopes Ferreira and João Canas Ferreira. An FPGA-Oriented Baseband Modulator Architecture for 4G/5G Communication Scenarios. *Electronics*, 8(1), 2019. ISSN 2079-9292. doi: 10.3390/electronics8010002.

F.L. Luo and C. Zhang. *Signal Processing for 5G: Algorithms and Implementations*. Wiley - IEEE. Wiley, 2016. ISBN 9781119116462.

J. C. Lyke, C. G. Christodoulou, G. A. Vera, and A. H. Edwards. An Introduction to Reconfigurable Systems. *Proceedings of the IEEE*, 103(3):291–317, March 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2015.2397832.

Patrick Marsch, Ömer Bulakci, Olav Queseth, and Mauro Boldi. *5G System Design: Architectural and Functional Considerations and Long Term Research*. John Wiley & Sons, 2018.

T. McCreary. On frequency sampling digital filters. *IEEE Transactions on Audio and Electroacoustics*, 20(3):222–223, August 1972. ISSN 0018-9278. doi: 10.1109/TAU.1972.1162373.

S. Medjkouh, J. Nadal, C. A. Nour, and A. Baghdadi. Reduced complexity FPGA implementation for UF-OFDM frequency domain transmitter. In *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2017. doi: 10.1109/SiPS.2017.8110013.

U. Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*. Signals and Communication Technology. Springer Berlin Heidelberg, 2007. ISBN 9783540726135.

J. Mitola. The software radio architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995. ISSN 0163-6804. doi: 10.1109/35.393001.

J. Mitola and G. Q. Maguire. Cognitive radio: making software radios more personal. *IEEE Personal Communications*, 6(4):13–18, Aug 1999. ISSN 1070-9916. doi: 10.1109/98.788210.

C. Moy and J. Palicot. Software radio: a catalyst for wireless innovation. *IEEE Communications Magazine*, 53(9):24–30, September 2015. ISSN 0163-6804. doi: 10.1109/MCOM.2015.7263342.

J. Nadal, C. Abdel Nour, and A. Baghdadi. Low-Complexity Pipelined Architecture for FBMC/OQAM Transmitter. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(1):19–23, Jan 2016. ISSN 1549-7747. doi: 10.1109/TCSII.2015.2468926.

J. Nadal, C. A. Nour, and A. Baghdadi. Flexible hardware platform for demonstrating new 5G waveform candidates. In *2017 29th International Conference on Microelectronics (ICM)*, Dec 2017. doi: 10.1109/ICM.2017.8268851.

Jeremy Nadal, Charbel Abdel Nour, and Amer Baghdadi. Flexible and efficient hardware platform and architectures for waveform design and proof-of-concept in the context of 5G. *AEU - International Journal of Electronics and Communications*, 97:85 – 93, 2018. ISSN 1434-8411. doi: 10.1016/j.aeue.2018.09.030.

A. Nimr, M. Chafii, and G. P. Fettweis. Unified Low Complexity Radix-2 Architectures for Time and Frequency-Domain GFDM Modem. *IEEE Circuits and Systems Magazine*, 18(4):18–31, Fourthquarter 2018. ISSN 1531-636X. doi: 10.1109/MCAS.2018.2872662.

J. Nunez-Yanez and A. Beldachi. Run-time power and performance scaling with CPU-FPGA hybrids. In *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 55–60, July 2014. doi: 10.1109/AHS.2014.6880158.

L. Orozco-Galvan, R. Parra-Michel, and E. Romero-Aguirre. Reconfigurable architecture based on FPGA for OFDM transmitter. In *2015 7th IEEE Latin-American Conference on Communications*, Nov 2015. doi: 10.1109/LATINCOM.2015.7430147.

Ju Hwa Pan, T. Mitra, and Weng-Fai Wong. Configuration bitstream compression for dynamically reconfigurable FPGAs. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pages 766–773, Nov 2004. doi: 10.1109/ICCAD.2004.1382679.

Kyprianos Papadimitriou, Apostolos Dollas, and Scott Hauck. Performance of Partial Reconfiguration in FPGA Systems: A Survey and a Cost Model. *ACM Trans. Reconfigurable Technol. Syst.*, 4(4):36:1–36:24, December 2011. ISSN 1936-7406.

I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai. A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions. *IEEE Communications Surveys Tutorials*, 20(4): 3098–3130, Fourthquarter 2018. ISSN 1553-877X. doi: 10.1109/COMST.2018.2841349.

Michel Pecot. Enabling High-Speed Radio Designs with Xilinx All Programmable FPGAs and SoCs. Technical Report WP445, Xilinx Inc., Jan 2014. v1.0.

L. Pezzarossa, A. T. Kristensen, M. Schoeberl, and J. Sparsø. Can real-time systems benefit from dynamic partial reconfiguration? In *2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pages 1–6, Oct 2017.

T. H. Pham, S. A. Fahmy, and I. V. McLoughlin. An End-to-End Multi-Standard OFDM Transceiver Architecture Using FPGA Partial Reconfiguration. *IEEE Access*, 5:21002–21015, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2756914.

A. Prata, R. F. Cordeiro, D. C. Dinis, A. S. R. Oliveira, J. Vieira, and N. B. Carvalho. All-digital transceivers — Recent advances and trends. In *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 233–236, Dec 2016. doi: 10.1109/ICECS.2016.7841175.

M. Rihani, M. Mroue, J. Prevotct, F. Nouvel, and Y. Mohanna. A Neural Network Based Handover for Multi-RAT Heterogeneous Networks with Learning Agent. In *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–6, July 2018. doi: 10.1109/ReCoSoC.2018.8449382.

Mohamad-Al-Fadl Rihani, Mohamad Mroue, Jean-Christophe Prévotet, Fabienne Nouvel, and Yasser Mohanna. ARM-FPGA-based platform for reconfigurable wireless communication systems using partial reconfiguration. *EURASIP Journal on Embedded Systems*, 2017(1):35, Dec 2017. doi: 10.1186/s13639-017-0083-9.

Bertrand Rousseau, Philippe Manet, Thibault Delavallée, Igor Loiselle, and Jean-Didier Legat. Dynamically Reconfigurable Architectures for Software-Defined Radio in Professional Electronic Applications. In Gabriela Nicolescu, Ian O'Connor, and Christian Piguet, editors, *Design Technology for Heterogeneous Embedded Systems*, pages 437–455. Springer Netherlands, 2012. ISBN 978-94-007-1125-9.

F. Schaich. Filterbank based multi carrier transmission (FBMC) - evolving OFDM: FBMC in the context of WiMAX. In *2010 European Wireless Conference (EW)*, pages 1051–1058, April 2010.

F. Schaich, T. Wild, and Y. Chen. Waveform Contenders for 5G - Suitability for Short Packet and Low Latency Transmissions. In *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, May 2014. doi: 10.1109/VTCSpring.2014.7023145.

M. Schellmann, Z. Zhao, H. Lin, P. Siohan, N. Rajatheva, V. Luecken, and A. Ishaque. FBMC-based air interface for 5G mobile: Challenges and proposed solutions. In *2014 9th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, pages 102–107, June 2014. doi: 10.4108/icst.crowncom.2014.255708.

S. Shreejith, B. Banarjee, K. Vipin, and S. A. Fahmy. Dynamic Cognitive Radios on the Xilinx Zynq Hybrid FPGA. In *Proceedings of the International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)*, April 2015.

M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis. 5G-Enabled Tactile Internet. *IEEE Journal on Selected Areas in Communications*, 34(3):460–473, March 2016. ISSN 0733-8716. doi: 10.1109/JSAC.2016.2525398.

P. Siohan, C. Siclet, and N. Lacaille. Analysis and design of OFDM/OQAM systems based on filterbank theory. *IEEE Transactions on Signal Processing*, 50(5):1170–1183, May 2002. ISSN 1053-587X. doi: 10.1109/78.995073.

D. Soldani, F. Fadini, H. Rasanen, J. Duran, T. Niemela, D. Chandramouli, T. Hoglund, K. Doppler, T. Himanen, J. Laiho, and N. Nanavaty. 5G Mobile Systems for Healthcare. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5, June 2017. doi: 10.1109/VTCSpring.2017.8108602.

L. Tang, J. A. Ambrose, and S. Parameswaran. Reconfigurable pipelined coprocessor for multi-mode communication transmission. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–8, May 2013.

Jim Tatsukawa. *XAPP888 - MMCM and PLL Dynamic Reconfiguration*. Xilinx Inc. v1.7.

R. Tessier, K. Pocek, and A. DeHon. Reconfigurable Computing Architectures. *Proceedings of the IEEE*, 103(3):332–354, March 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2386883.

LH Thomas. Using a computer to solve problems in physics. In W.F. Freiberger and W. Prager, editors, *Applications of digital computers*, pages 44–45. Ginn, 1963.

S. M. Trimberger. Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology. *Proceedings of the IEEE*, 103(3):318–331, March 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2015.2392104.

3GPP TS. NR; NR and NG-RAN Overall Description; Stage 2 (Release 15). http://www.3gpp.org/DynaReport/38-series.htm, October 2018. 38.300 V15.3.1.

Pei-Yun Tsai, Tsung-Hsueh Lee, and Tzi-Dar Chiueh. Power-Efficient Continuous-Flow Memory-Based FFT Processor for WiMax OFDM Mode. In *ISPACS '06. International Symposium on Intelligent Signal Processing and Communications, 2006.*, pages 622–625, Dec 2006. doi: 10.1109/ISPACS.2006.364733.

V. Vakilian, T. Wild, F. Schaich, S. ten Brink, and J. Frigon. Universal-filtered multi-carrier technique for wireless systems beyond LTE. In *2013 IEEE Globecom Workshops*, pages 223–228, Dec 2013. doi: 10.1109/GLOCOMW.2013.6824990.

J. J. van de Beek, M. Sandell, and P. O. Borjesson. ML estimation of time and frequency offset in OFDM systems. *IEEE Transactions on Signal Processing*, 45(7):1800–1805, July 1997. ISSN 1053-587X. doi: 10.1109/78.599949.

L. Varga and Z. Kollár. Low complexity FBMC transceiver for FPGA implementation. In *2013 23rd International Conference Radioelektronika*, pages 219–223, April 2013.

A. Viholainen, M. Bellanger, and M. Huchard. D5.1 Prototype filter and structure optimization. Technical report, PHYDYAS project, 2009.

K. Vipin and S. A. Fahmy. Mapping adaptive hardware systems with partial reconfiguration using CoPR for Zynq. In *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 1–8, June 2015. doi: 10.1109/AHS.2015.7231169.

K. Vipin and S.A. Fahmy. ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq. *IEEE Embedded Systems Letters*, 6(3):41–44, Sept 2014. ISSN 1943-0663. doi: 10.1109/LES.2014.2314390.

Kizheppatt Vipin and Suhaib A. Fahmy. FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications. *ACM Comput. Surv.*, 51(4):72:1–72:39, July 2018. ISSN 0360-0300. doi: 10.1145/3193827.

X. Wang, T. Wild, F. Schaich, and A. Fonseca dos Santos. Universal Filtered Multi-Carrier with Leakage-Based Filter Optimization. In *European Wireless 2014; 20th European Wireless Conference*, pages 1–5, May 2014.

P. Weitkemper, J. Koppenborg, J. Bazzi, R. Rheinschmitt, K. Kusume, D. Samardzija, R. Fuchs, and A. Benjebbour. Hardware experiments on multi-carrier waveforms for 5G. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6, April 2016.

Shmuel Winograd. On computing the discrete Fourier transform. *Mathematics of computation*, 32 (141):175–199, 1978.

M. J. Wirthlin and B. L. Hutchings. Improving functional density using run-time circuit reconfiguration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(2):247–256, June 1998.

M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, March 2017. ISSN 1932-4529. doi: 10.1109/MIE.2017.2649104.

YongJu Won, JongGyu Oh, JinSeop Lee, and JoonTae Kim. A Study of an Iterative Channel Estimation Scheme of FS-FBMC System, 2017. DOI: 10.1155/2017/6784142.

J. Wu, Z. Zhang, Y. Hong, and Y. Wen. Cloud radio access network (C-RAN): a primer. *IEEE Network*, 29(1):35–41, Jan 2015. ISSN 0890-8044. doi: 10.1109/MNET.2015.7018201.

X. Xiao, E. Oruklu, and J. Saniie. An Efficient FFT Engine With Reduced Addressing Logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(11):1149–1153, Nov 2008a. ISSN 1549-7747. doi: 10.1109/TCSII.2008.2004540.

X. Xiao, E. Oruklu, and J. Saniie. An Efficient FFT Engine With Reduced Addressing Logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(11):1149–1153, Nov 2008b. ISSN 1549-7747. doi: 10.1109/TCSII.2008.2004540.

UG472 - 7 Series FPGA Clocking Resources User Guide. Xilinx Inc., a. v1.13.

UG585 - Zynq-7000 Technical Reference Manual. Xilinx Inc., b. v1.11.

UG954 - ZC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable SoC User Guide. Xilinx Inc., c. v1.6.

UG901 - Vivado Design Suite User Guide: Synthesis. Xilinx Inc., June 2015a.

UG909 - Vivado Design Suite User Guide: Partial Reconfiguration. Xilinx Inc., June 2015b.

PG151 - Divider Generator v5.1 LogiCORE IP Product Guide. Xilinx Inc., October 2016a.

UG953 - Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide. Xilinx Inc., November 2016b.

UG470 - 7 Series FPGAs Configuration User Guide. Xilinx Inc., August 2018.

G. Yuan, X. Zhang, W. Wang, and Y. Yang. Carrier aggregation for LTE-advanced mobile communication systems. *IEEE Communications Magazine*, 48(2):88–93, February 2010. ISSN 0163-6804. doi: 10.1109/MCOM.2010.5402669.

A. A. Zaidi, R. Baldemair, H. Tullberg, H. Bjorkegren, L. Sundstrom, J. Medbo, C. Kilinc, and I. Da Silva. Waveform and Numerology to Support 5G Services and Requirements. *IEEE Communications Magazine*, 54(11):90–98, November 2016. ISSN 0163-6804. doi: 10.1109/MCOM.2016.1600336CM.

Ali Zaidi, Fredrik Athley, Jonas Medbo, Ulf Gustavsson, Giuseppe Durisi, and Xiaoming Chen. Chapter 5 - Multicarrier Waveforms. In Ali Zaidi, Fredrik Athley, Jonas Medbo, Ulf Gustavsson, Giuseppe Durisi, and Xiaoming Chen, editors, *5G Physical Layer*, pages 119 – 158. Academic Press, 2018. ISBN 978-0-12-814578-4. doi: https://doi.org/10.1016/B978-0-12-814578-4.00010-2.

B. Zhang and X. Guo. A novel reconfigurable architecture for generic OFDM modulator based on FPGA. In *16th International Conference on Advanced Communication Technology*, pages 851–854, Feb 2014. doi: 10.1109/ICACT.2014.6779080.

Chenxin Zhang, Liang Liu, and Viktor Öwall. *Heterogeneous Reconfigurable Processors for Real-Time Baseband Processing*. Springer International Publishing, Cham, 2016. ISBN 978-3-319-24002-2 978-3-319-24004-6.

Q. Zhao and B. M. Sadler. A Survey of Dynamic Spectrum Access. *IEEE Signal Processing Magazine*, 24(3):79–89, May 2007. ISSN 1053-5888. doi: 10.1109/MSP.2007.361604.